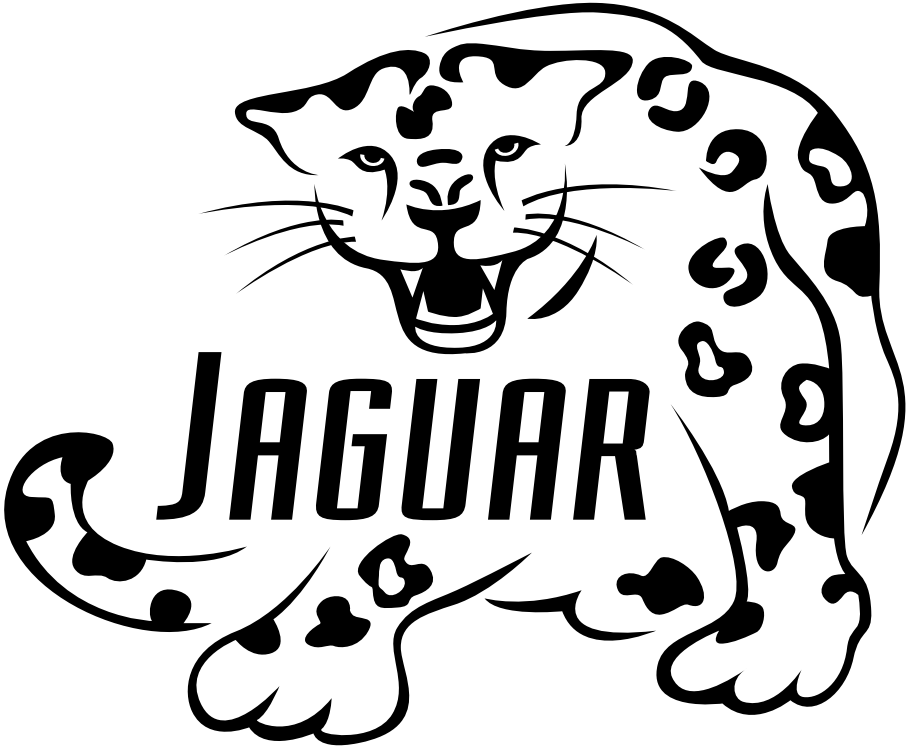
# JAGUAR USER'S GUIDE

**Version 4.2**

**March 2002**

**Julie R. Wright**




**© 1998, 2000, 2001, 2002 Schrödinger, Inc.**
**All Rights Reserved.**

# JAGUAR USER'S GUIDE

**Version 4.2**

**March 2002**

**Julie R. Wright**

Schrödinger, Inc.
1500 SW First Avenue, Suite 1180
Portland, OR 97201
Telephone: (503) 299-1150
Fax: (503) 299-4532
Email: help@schrodinger.com

© 1998, 2000, 2001, 2002 Schrödinger, Inc.
All rights reserved.

Copyright © 1998, 2000, 2001, 2002 Schrödinger, Inc.
All rights reserved.
Schrödinger, Inc. provides this publication “as is” without warranty of any kind, either expressed or implied.

NBO 4.0 is copyright © 1996 Board of Regents of the University of Wisconsin System on behalf of the Theoretical Chemistry Institute.

Babel 1.3 is copyright © 1992-96 W. Patrick Walters and Matthew T. Stahl.

SPARTAN is a trademark owned by Wavefunction, Inc.

BIOGRAF and Cerius$^2$ are trademarks of Molecular Simulations, Inc.

Gaussian, Gaussian 90, Gaussian 92, and Gaussian 94 are federally registered trademarks of Gaussian, Inc.

Silicon Graphics, IRIX, and OpenGL are trademarks of Silicon Graphics, Inc.

IBM is a registered trademark of International Business Machines Incorporated.

DEC is a trademark of Digital Equipment Corporation.

UNIX is a registered trademark of UNIX Systems Laboratories, Inc.

X Window System is a registered trademark of Massachusetts Institute of Technology.

All other brand or product names are trademarks or registered trademarks of their respective companies or organizations.

*420067032002*

# Table of Contents

1. Introduction ..... 1
1.1. Overview of this User's Guide ..... 1
1.2. Citing Jaguar in Publications ..... 3
1.3. Technical Support ..... 3
2. Using Jaguar's Interface ..... 4
2.1. Sample Calculation ..... 5
2.2. Molecular Structure Input ..... 8
Inputting or Editing a Geometry Within the Interface ..... 8
Cartesian Format for Geometry Input ..... 9
Variables in Cartesian Input ..... 10
Constraining Cartesian Coordinates ..... 10
Z-Matrix Format for Geometry Input ..... 11
Variables and Dummy Atoms in Z-Matrix Input ..... 13
Constraining Z-Matrix Bond Lengths or Angles ..... 14
Counterpoise Calculations ..... 15
Specifying Coordinates for Hessian Refinement ..... 15
2.3. Charge and Multiplicity (State) ..... 16
2.4. Reading Files ..... 16
Reading a Geometry, But No Calculation Settings ..... 17
Reading In Both Geometries and Job Settings ..... 19
Read as Geometry 2 or Geometry 3 Settings ..... 19
2.5. Geometry Display ..... 19
Display Window Basics: Orientation & Mouse Control ..... 20
Display Styles ..... 21
Labels ..... 22
Other Display Options ..... 22
Choosing a Structure To Display ..... 23
Closing the Display Window ..... 23
2.6. Cleaning up Molecular Geometries ..... 23
The Cleanup Button ..... 23
The Symmetrize Molecule Button ..... 24
Tolerance ..... 25
Finding the Point Group ..... 25
Symmetrizing Coordinates ..... 26
2.7. Running Jobs and Saving Input ..... 26
Starting Individual Jobs from the Interface ..... 26
Running Batch Jobs or Scripts from the Interface ..... 29
Saving Input Files ..... 31
Output ..... 32
2.8. Other Interface Features ..... 32

Checking Jobs with the Job Status Window ..... 33
Resetting and Quitting ..... 33
About and Help Buttons ..... 34
Editing Input ..... 34
Other Main Window Options ..... 35

3. Options ..... 37

3.1. Density Functional Theory (DFT) Settings ..... 38
Stage and Grid Density ..... 38
Method Options ..... 39
Functionals ..... 40

3.2. Local MP2 Settings ..... 41
Summary of the LMP2 Method in Jaguar ..... 42
Setting Up an LMP2 Calculation ..... 43

3.3. Generalized Valence Bond (GVB) Settings ..... 44
GVB or GVB-RCI Pair Input ..... 44

3.4. GVB-LMP2 Calculations ..... 45

3.5. Solvation ..... 46
Solvent Parameters ..... 47
Performing or Skipping a Gas Phase Optimization ..... 48

3.6. Properties ..... 48
Electrostatic Potential Fitting ..... 49
Multipole Moments ..... 50
Polarizability and Hyperpolarizability ..... 50
Electron Density ..... 51
Mulliken Population Analysis ..... 51
Natural Bond Orbital (NBO) Analysis ..... 52

3.7. Frequencies & Related Properties ..... 52
Frequencies ..... 53
Atomic Masses ..... 53
Scaling of Frequencies ..... 54
Infrared Intensities ..... 54
Thermochemical Properties ..... 55

3.8. Basis Set ..... 56

3.9. Methods ..... 60
Wavefunction Type (Restricted or Unrestricted) ..... 60
Choosing an Initial Guess Type ..... 61
Convergence Issues ..... 62
Accuracy Level ..... 63
Analytic Corrections ..... 64
Final Localization of the Orbitals ..... 64
Symmetry ..... 65

4. Optimizations and Scans ..... 66
4.1. Geometry Optimization: The Basics ..... 66
Maximum Iterations (Number of Structures Generated) ..... 67
Geometry Convergence Issues ..... 67
The Initial Hessian ..... 68
Trust Radius ..... 69
4.2. Constraining Coordinates ..... 69
Freezing All Bond Lengths, Bond Angles, or Torsional Angles ..... 69
Freezing Specific Coordinates ..... 70
Applying Constraints by Using Variables ..... 71
4.3. Transition State Optimizations ..... 72
Transition State Search Method ..... 72
Specifying Different Structures for the Reaction ..... 73
Initial LST Guess ..... 74
Searching Along a Particular Path or Eigenvector ..... 75
Eigenvector Following ..... 76
Refinement of the Initial Hessian ..... 76
Specifying Coordinates for Hessian Refinement ..... 77
4.4. Geometry Scans ..... 78

5. Output ..... 80
5.1. Summarizing Jaguar Results ..... 80
Reporting Final Results from One or More Jobs ..... 81
Reporting Intermediate Results ..... 83
Reporting Results for Each Atom ..... 84
5.2. Output from a Standard HF Calculation ..... 85
5.3. Output File Changes for Calculation Options ..... 90
DFT ..... 90
LMP2 ..... 91
GVB ..... 91
GVB-RCI ..... 93
Geometry or Transition State Optimization (HF, GVB, DFT, and LMP2) ..... 93
Optimizations with GVB-RCI Wavefunctions ..... 97
Solvation ..... 98
Geometry Optimization in Solution ..... 102
Properties ..... 102
Frequency, IR Intensity, and Thermochemistry Output ..... 106
Basis Set ..... 108
Methods ..... 108
5.4. Standard Output Options ..... 108
*echo input file and parameter list* ..... 108
*memory, disk, and i/o information* ..... 109
*detailed timing information* ..... 109
*geometries in atomic units also* ..... 109

*multipole moments in atomic units also* ..... 109
*bond lengths and angles* ..... 109
*connectivity table* ..... 110
*overlap matrix* ..... 110
*geometry optimization details* ..... 110
*one-electron Hamiltonian* ..... 110
*Gaussian function list (basis set)* ..... 111
*Gaussian function list (derivatives)* ..... 114
5.5. Files Output Options ..... 114
*Gaussian-92 input deck (.g92)* ..... 114
*GAMESS input file (.gamess)* ..... 115
*SPARTAN archive file (.arc)* ..... 115
*Gaussian-92 basis set (.gbs)* ..... 115
*XYZ file (.xyz)* ..... 115
Molden orbitals file (.molf) ..... 115
5.6. Output Options Per Iteration ..... 116
*energy components* ..... 116
*density matrix* ..... 116
*All J and K matrices, AO space* ..... 117
*Fock matrix in AO (HF) or MO (GVB) space* ..... 117
*Fock matrix in CO space* ..... 117
*GVB data: f, a, b, ci coefficients, etc.* ..... 117
5.7. Output Options for Orbitals ..... 118
When: ..... 118
What: ..... 119
How: ..... 119
5.8. The Log File ..... 122

6. Tips and Suggestions ..... 124
6.1. Customizing Interface Settings: The jaguar.hosts File ..... 124
6.2. Customizing the Interface Appearance ..... 127
6.3. Tips for Various Types of Jobs ..... 127
Organometallics and Other Difficult-to-Converge Systems ..... 128
GVB Calculations: GVB Pair Selection ..... 130
Geometry Optimization ..... 130
Electrostatic Potential Charge Fitting ..... 131
6.4. Restarting Jobs and Using Previous Results ..... 131
Finding the Restart File in the Temp Directory ..... 132
6.5. Using Jaguar with MOPAC ..... 133
Running MOPAC ..... 133
Adding MOPAC Geometries to Jaguar Input ..... 134
Using MOPAC Hessians in Jaguar Input ..... 134
Specifying Different File Names ..... 135
6.6. Suggestions for GAUSSIAN Users ..... 136

Generating GAUSSIAN Input Files with Jaguar ..... 136
Getting Basis Sets or Orbitals for GAUSSIAN ..... 138
Using GAUSSIAN Files as Jaguar Input ..... 138

7. Theory ..... 139
7.1. The Pseudospectral Method ..... 139
7.2. Pseudospectral Implementation of the GVB Method ..... 142
7.3. GVB-RCI Wavefunctions ..... 146
7.4. Pseudospectral Local MP2 Techniques ..... 148
7.5. Density Functional Theory ..... 152

8. The Jaguar Input File ..... 154
8.1. The jaguar Command ..... 154
Selecting a Calculation Host ..... 155
Selecting Particular Jaguar Executables ..... 156
Running a Jaguar Job from the Command Line ..... 156
Killing a Jaguar Job ..... 158
8.2. General Description of the Input File ..... 159
Sections Describing the Molecule and Calculation ..... 160
8.3. The zmat, zmat2, and zmat3 Sections ..... 162
8.4. The zvar, zvar2, and zvar3 Sections ..... 164
8.5. The coord and connect Sections ..... 165
8.6. The gen Section ..... 166
Geometry Input Keywords ..... 167
Molecular State Keywords (Charge and Multiplicity) ..... 167
Atomic Mass Keyword ..... 167
Symmetry-Related Keywords ..... 168
GVB and Lewis Dot Structure Keywords ..... 169
LMP2 Keywords ..... 171
DFT Keywords ..... 173
Geometry Optimization and Transition State Keywords ..... 177
Solvation Keywords ..... 183
Properties Keywords ..... 185
Frequency-Related Keywords ..... 188
Basis Set Keywords ..... 190
Keywords for SCF Methods ..... 191
Initial Guess Keywords ..... 195
Localization Keywords ..... 197
Using Babel to Output Files ..... 199
Standard Output Keywords ..... 201
Files Output Keywords ..... 203
Output Keywords for Each Iteration ..... 204
Orbital Output Keywords ..... 205
Grid and Dealiasing Function Keywords ..... 207

Memory Usage Keywords ..... 209
8.7. The gvb Section ..... 212
8.8. The lmp2 Section ..... 213
8.9. The atomic Section ..... 214
General Format of the atomic Section ..... 214
Input Types That Specify Physical Properties ..... 216
Basis, Grid, Dealiasing Function, and Charge Usage for Individual Atoms ..... 217
Defining Fragments ..... 218
8.10. The hess Section ..... 219
8.11. The guess Section ..... 220
8.12. The pointch Section ..... 222
8.13. The efields Section ..... 222
8.14. The ham Section ..... 223
8.15. The orbman Section ..... 223
8.16. The echo Section ..... 225
8.17. The path Section ..... 225
8.18. The plot Section ..... 228
8.19. NBO Sections ..... 229
8.20. Running Multiple Jobs: jaguar batch ..... 230
Batch Input File Format ..... 230
Batch Input File Example ..... 233
Running "jaguar batch" ..... 234

9. Other Jaguar Files ..... 237
9.1. The Basis Set File ..... 238
9.2. The Initial Guess Data File ..... 242
9.3. The Dealiasing Function File ..... 244
File Format and Description ..... 245
Sample File ..... 247
9.4. The Grid File ..... 249
File Format and Description ..... 249
9.5. The Cutoff File ..... 253
9.6. The Lewis File ..... 255
Describing Bonding Types in the Lewis File ..... 257
Describing Hybridization Types in the Lewis File ..... 258
Setting van der Waals Radii from Lewis File Data ..... 260
Default Behavior for Setting Radii ..... 264

10. Troubleshooting ..... 266
10.1. Problems Getting Started ..... 266
The SCHRODINGER Environment Variable ..... 267
Including the jaguar Command in Your Path ..... 268

Problems Starting the Interface ..... 268
Problems Related to Your Temporary Directory ..... 270
Problems Running Jaguar Calculations on Other Nodes ..... 271
10.2. On-Line Help ..... 273
10.3. Other Problems ..... 273

11. The Parallel Jaguar Module ..... 277
11.1. About Parallel Jaguar ..... 277
11.2. Installing Parallel Jaguar ..... 277
11.3. Requirements for Different Computer Platforms ..... 278
SGI ..... 278
LINUX ..... 279
IBM ..... 283
HP, Compaq, Sun ..... 284
11.4. Running Jobs In Parallel ..... 285

12. The pKa Prediction Module ..... 289
12.1. Introduction ..... 289
12.2. Theory of pKa Calculation ..... 291
Ab initio Quantum Chemical Calculation of pKa's ..... 291
Empirical Corrections ..... 294
12.3. Predicting pKa's in Complex Systems ..... 295
Overview ..... 295
Conformational Flexibility ..... 295
Equivalent Sites ..... 297
Multiple Protonation Sites ..... 297
12.4. Results ..... 298
12.5. Guide to Running the Program ..... 308
Installing the pKa Module ..... 308
JAGUAR Input Files for pKa Calculations ..... 308
Running pKa Calculations ..... 309
Monitoring pKa Calculations ..... 310
Initial Geometry ..... 310

Information by Element ..... 313

References ..... 321

Index ..... 331

Keywords ..... 347

# 1. Introduction

## 1.1. Overview of this User's Guide

The Jaguar User's Guide is intended to help you perform ab initio calculations for a variety of methods, parameters, and calculated properties. On-line help is also available within Jaguar's graphical interface, although the information in this User's Guide is generally more comprehensive.

Chapter 2 contains information you will need to run Jaguar, including information about using the graphical user interface, geometry input formats, specifying file names for input and output, displaying molecular geometries, symmetrizing geometries, and setting run-time parameters such as the machine the calculation will be performed upon. We suggest that you start by trying the sample calculation in section 2.1. If the calculation runs successfully, you can proceed to the rest of the chapter to learn how to input molecular structures and run jobs. If you have problems starting the interface or running the sample calculation, see the troubleshooting information in section 10.1.

Chapter 3 and Chapter 4 describe the available calculation options, allowing you to specify what properties you wish the program to calculate and what methods you wish it to use. Chapter 3 includes information on using generalized valence bond (GVB), restricted configuration interaction (RCI), Møller-Plesset second-order perturbation theory, and density functional theory (DFT) techniques; calculating solvation energies, vibrational frequencies, hyperpolarizabilities, multipole moments, and other properties; fitting charges; specifying basis sets; and various other options. Chapter 4

describes optimizations of the molecular structure, transition state searches, and geometry scans.

Chapter 5 describes how to summarize Jaguar output and the output or printing options available from the interface. The output file containing the primary Jaguar output is first described for cases where no Output options have been selected. Next, the output given when various Output settings are turned on is explained. Finally, the log file, another output file, is described.

The next chapter contains tips and suggestions for using Jaguar. The information in Chapter 6 includes a description of how to customize the interface; some general tips for different sorts of calculations; a description of how to restart calculations, how to incorporate results from previous runs; and some tips for people who use both Jaguar and GAUSSIAN.

Chapter 7 contains a description of some of the theory behind the pseudospectral method and the electron correlation methods used in Jaguar. This chapter includes information on pseudospectral implementations of GVB, GVB-RCI, and local MP2 techniques, and a brief description of density functional theory.

Chapter 8 describes the Jaguar input file in detail. You may find this chapter especially useful if you want to run some jobs without using the interface. Chapter 9 describes other Jaguar files which are necessary for calculations. *Neither Chapter 8 nor Chapter 9 is necessary reading if you wish to run all jobs from the interface*, but you may want to skim them anyway to find out more about Jaguar and the methods it uses.

Chapter 10 contains some troubleshooting hints concerning the various sorts of problems you may encounter, especially when first setting up Jaguar on your system.

Throughout the User's Guide, numbers in brackets, like this: [13], refer to sources you may want to consult. The full reference list appears at the end of this manual. Superscripts in the text correspond to footnotes that list the Jaguar input file entries which correspond to particular interface settings. You can ignore the footnotes if you like, but you may find them useful for setting up files to do runs without using the interface, or for interpreting the input file.

## 1.2. CITING JAGUAR IN PUBLICATIONS

Use of this program should be acknowledged in publications as:

Jaguar 4.2, Schrödinger, Inc., Portland, OR, 1991-2000.

## 1.3. TECHNICAL SUPPORT

If you have questions or problems concerning Jaguar, please report them to:

Schrödinger, Inc.
1500 SW First Avenue, Suite 1180
Portland, OR 97201
Telephone: (503) 299-1150
Fax: (503) 299-4532
E-mail: help@schrodinger.com

We recommend contacting us by e-mail. When corresponding, please include relevant input and output. This information will help us evaluate problems more quickly.

# 2. Using Jaguar's Interface

Jaguar has an X Window System (or "X") graphical interface in order to simplify the submission of jobs. You can run the interface and the actual Jaguar calculation on different machines. In addition, as with any X program, the machine running the interface (the X client) does not need to be the machine or terminal which displays the interface (the X server). Thus, from any X terminal or workstation running X, you can log onto a machine where the Jaguar interface is installed, and submit jobs on yet another machine on which the Jaguar main executables are installed.

Without the graphical interface, you would have to create input files with particular formats in order to run Jaguar. The graphical interface creates these input files for you, based on the information you give it, and submits the job, thus freeing you from learning the input format and program sequences and instead allowing you to concentrate on the science involved. The interface also provides a convenient method of incorporating other data, such as molecular geometries produced by modeling packages.

Try the sample calculation in Section 2.1 in order to get some experience with running Jaguar and to make sure your system is properly set up. If you have problems starting or using the interface or performing the calculation, you may be able to solve them using the troubleshooting suggestions in section 10.1. If any problems persist, please contact your system manager or Schrödinger, Inc.

The rest of this chapter describes the basics of using the interface, including inputting a geometry and submitting a job. The footnotes in this chapter describe Jaguar input file keywords and sections that correspond to particular interface settings. If you are working from

the interface, you can ignore these footnotes, but you may later find them helpful if you decide to use input files to submit jobs without using the interface or if you want to edit keywords directly by using the Edit Job window described in Section 2.8 under *Editing Input*.

# 2.1. SAMPLE CALCULATION

The brief sample calculation suggested in this section will only work if Jaguar has already been correctly installed. If the calculation does not work, try the suggestions in section 10.1, or see your system manager or the person who installed Jaguar at your site. Please contact Schrödinger if you cannot resolve the installation problems.

First, from a terminal or workstation running the X Window System, log into a machine where the Jaguar interface is installed. From a directory where you want the Jaguar input and output files for the sample job to appear, enter the command

```
jaguar
```

The Jaguar main window and the Display window should appear. If the windows appear as outlines, or if you wish to alter them, you can use the mouse to position and size them. The Jaguar main window contains buttons that let you access various interface features. When you input a molecular structure, the Display window will show it. For now, if you want to, you can temporarily remove the Display window by dragging the mouse over its File menu and releasing it over the Close option.

Next, you should input a molecular geometry (structure). From the Jaguar main window, click on the button marked Edit near the Geometry heading. The Edit Geometry window should open. Move the mouse into the blank panel in the middle of the window, and enter the following lines, which will provide Jaguar with the geometry of a water molecule:

```
O    0.0         0.0   -0.1135016
H1   0.753108    0.0    0.4540064
H2  -0.753108    0.0    0.4540064
```

The labels begin with element symbols, 'O' and 'H.' Extra characters, a '1' and a '2,' have been added to the hydrogen labels to distinguish between the atoms. The next three numbers on each line give the x, y, and z Cartesian coordinates of the atoms in the geometry, in Angstroms. The number of spacing characters does not matter, as long as you use at least one to separate different items. Make sure to

hit <return> at the end of each line, however. When you have finished entering the water geometry, click File in the upper left corner of the Edit Geometry window and select Save to save your changes, then click File again, selecting Close to close the Edit Geometry window.

The molecular structure should now be shown in the Display window. If you closed this window earlier, you can reopen it by clicking on the Display button, which appears near the Geometry heading in the main interface window. Section 2.5 discusses the Display feature in more detail.

Give your job a name by entering a single word in the Job Name box in the main Jaguar window. The names of the input, output, and log files for your job will depend on your entry: the Jaguar input file will be named *jobname*.in, the output file will be named *jobname*.out, and the log file will be named *jobname*.log, where *jobname* is your Job Name entry.

If you want to, you can now use the buttons in the bottom panel of the main Jaguar window to open other windows (the DFT window, for example), which you can use to set up the calculation you want to perform. These possible selections are described in Chapter 3 of this manual. If you don't make any non-default settings in these windows, Jaguar will run a single-point Hartree-Fock calculations ("single-point" means using only the molecular structure in the input, without optimizing that structure). Any time you want to retract any of the settings you have made since pulling up a window, you can hit the Cancel button at the bottom of the window. If, instead, you hit OK, the settings are registered in whatever way is appropriate for that window.

You could also now use the buttons near the Output heading to make selections requesting extra output for the job, but we suggest running this trial calculation without requesting any non-default output. Chapter 5 of this manual describes the output options.

When you have finished setting up your calculation, click on the Run button near the Jobs heading in the main interface window. The Run window will appear. The calculation host, the machine upon which the job will actually run, is listed at the top of the window. If Jaguar is installed on more than one machine at your site, you should be able to change the choice of calculation host by clicking on the host name shown and dragging the mouse to another name in the list. The Temp directory selection is a directory on the calculation host which will be used during the calculation to store temporary files. You should check

from your X window to make sure the temp directory already exists on the calculation host; if it does not, you should create it.

The Run window settings should now be fine as they are, so you can start running the job by clicking on the RUN button at the bottom of the window. (If you do not want to perform the sample run using the selections shown, you can find more information on the Run window in Section 2.7. Section 6.1 explains how to change which options are presented in the Run window.)

After you have started running the job by clicking RUN in the Run window, another window will appear summarizing information about the job and where it is running. After you click the OK button in this window, it and the Run window will close and another window will open, possibly showing the Jaguar logo at first. This new window, the Job Status window, will automatically update to show your job's progress. As each separate program in the Jaguar code finishes running, its completion is noted in the window. When the program scf is running, the Job Status window also displays the energy and other data of each iteration. See section 5.8 on the log file if you wish to see more information on this data. You can close the Job Status window by clicking on the Close button at the upper right of the window; if you later want to reopen it, you can do so by clicking the Check button, which appears in the main window near the Jobs heading.

When the job finishes running, its output file will be copied to the directory where you started the interface. The output file's name always begins with the same characters as the job name you entered earlier, and ends with the extension ".out." For instance, if you entered the job name "h2o," the output file is called "h2o.out." You can look at the output file from your X window. If you want to exit the Jaguar interface, you can select the Quit button from near the top of the main interface window.

If you are satisfied with the results of this sample run, you should continue reading this chapter to learn more about using the interface. If you were unable to run the sample calculation, try following the troubleshooting suggestions in section 10.1. If you wish to change the options shown in the Run window, you can read section 6.1 for a description of how to do so, although you should probably wait until you have read the rest of this chapter and learned more about the interface.

# 2.2. MOLECULAR STRUCTURE INPUT

After you have started the interface by typing "jaguar" or "jaguar &" at a UNIX prompt on an X terminal or workstation, the first thing you will probably want to do for any Jaguar calculation is to input a molecular structure (geometry).[1] You can either use the interface to read in a file in one of several types of formats, as described in Section 2.4, or you can input and edit geometry coordinates yourself through the interface. This section describes how to create or edit a geometry and the input formats for Cartesian and Z-matrix geometries.

The geometry input also controls constraints of bond lengths or angles for geometry optimization and allows you to specify atoms for a counterpoise calculation. These aspects of geometry input are explained in this section as well.

## *Inputting or Editing a Geometry Within the Interface*

To input or edit a geometry by hand (or to just look at the coordinates), select Edit from near the Geometry heading. The Edit window, a text window, will appear. If you have not yet used the interface to read in a geometry file, you may enter the geometry there, either by cutting and pasting from another text window, as described in your X documentation, or by simply typing in the geometry. You can also use the Edit window to change a geometry which you input earlier.

Your geometry can be in Cartesian (x,y,z) coordinates or in Z-matrix format. These formats are described below. The Cut, Copy, and Paste options in the Edit menu, which you can access by clicking on Edit (in the bar near the top of the window), may be useful to you for geometry input. To remove the entire geometry input at any point, click on Edit and select Clear.

You can also alter the geometry input by using the Z-matrix pull-down window. You can use the Convert to Z-matrix and Convert to Cartesians options to switch between Z-matrix format and Cartesian format. (These formats are described in more detail later in this section.) The option Assign standard atom labels converts all atom labels to the form *El#*, where *El* is the standard elemental symbol (Fe for iron, for instance) and *#* is a number indicating the line of the

1. If you were working directly from an input file without using the interface, the geometry input would be in the **zmat** and **zvar** sections of the input file.

structure input on which the atom's coordinates are set (1 for the first atom, 2 for the second, and so on).

To save or remove your changes, or to close the Edit window, use the File pull-down menu. The Save option registers the changed geometry but leaves the window open. If you select Close, the window will close; if there are any unsaved changes, you will be asked if you want to save them or not. The Revert option lets you return to the original geometry (if any) in the window when you opened it, and Cancel closes the window without retaining any changes you have made from within it since you opened it.

The options in the Structure pull-down menu, and the Use initial geometry Z-matrix option in the Z-matrix pull-down menu, are useful for certain types of transition state optimization jobs, but not for other Jaguar jobs. Therefore, they will be described in section 4.3, which explains special options for transition state optimizations.

Please note that when you are editing a geometry and have not yet saved it, if you try to run a job or save an input file by hitting RUN in the Run window or OK in the Save window, you will get a warning about the open editing box. If you ignore the warning and proceed, the last geometry saved will be used, instead of the edited version.

## *Cartesian Format for Geometry Input*

The Cartesian geometry input format can simply consist of a list of atom labels and the atomic coordinates in Angstroms in Cartesian (x,y,z) form. For example, the input

```
O   0.000000    0.000000   -0.113502
H1  0.000000    0.753108    0.454006
H2  0.000000   -0.753108    0.454006
```

describes a water molecule. Each atomic label must start with the one- or two-letter elemental symbol given in the periodic table and may be followed by additional alphanumeric characters, as long as the atomic label has four or fewer characters and the atomic symbol remains clear—for instance, "HE5" would be interpreted as helium atom "5", not hydrogen atom "E5". Either lowercase or uppercase characters are allowed in atom labels. The coordinates may be specified in any valid C format, but each line of the geometry input should contain 80 characters or fewer.

## *Variables in Cartesian Input*

Coordinates can also be specified as variables whose values are set beneath the list of atomic coordinates. This feature makes it easier to input equal values and also makes it possible to keep several atoms within the same plane during a geometry optimization even while the coordinates change from their original values.

To use variables, type the variable name (zcoor, for instance) where you would normally type the corresponding numerical value for each relevant coordinate. You can put a + or – sign immediately before any variable, and you may use several variables if you want to. When you have entered the full geometry, add one or more lines setting the variables.

For instance, the Cartesian input

```
O   0.000000     0.000000  -0.113502
H1  0.000000     ycoor      zcoor
H2  0.000000    -ycoor      zcoor
ycoor=0.753108   zcoor=0.454006
```

describes the same water coordinates as the previous Cartesian input example. If you performed a geometry optimization using this input structure, its ycoor and zcoor values might change, but their values for one hydrogen atom would always be the same as those for the other hydrogen atom, so the H atoms in the final geometry would be in the same xy plane as each other.

At times, the Edit window may list the variable settings on several lines, preceded by a line saying "Z-variables". This format is just another way the interface has of separating the variable settings from the atomic coordinates.

One final note: whenever Cartesian input with variables is used for an optimization, Jaguar performs the optimization using Cartesian coordinates rather than generating redundant internal coordinates, and the optimization does not make use of molecular symmetry.

## *Constraining Cartesian Coordinates*

As described in the subsection *Variables in Cartesian Input*, you can force certain Cartesian coordinates to remain the same as each other during an optimization by using variables. You can also constrain any geometry in Cartesian format so that some of the Cartesian coordinates you specify are held fixed during a geometry optimization by adding a '#' sign after the appropriate coordinate or coordinates. For

example, if you added constraints to the zcoor variables in the water input example given above, as listed here:

```
O   0.000000     0.000000  -0.113502
H1  0.000000     ycoor      zcoor#
H2  0.000000    -ycoor      zcoor#
ycoor=0.753108   zcoor=0.454006
```

and performed a geometry optimization on this molecule, the H atoms would only be allowed to move within the xy plane in which they started.

If frozen Cartesian coordinates are included in the input for an optimization, Jaguar uses Cartesian coordinates for the optimization rather than generating redundant internal coordinates, and the optimization does not make use of molecular symmetry.

## *Z-Matrix Format for Geometry Input*

Like Cartesian input geometries, Z-matrix-format geometries should specify atoms by atom labels that begin with the one- or two-letter elemental symbol given in the periodic table, in either uppercase or lowercase characters. The elemental symbol may be followed by additional characters, as long as the atom label has four or fewer characters and the elemental symbol is still clear. Geometry input files in Z-matrix form can also contain comment lines beginning with pound signs (#), although comments should not be included in input entered in the interface's Edit window.

The first line of the Z-matrix should contain only one item: the atom label for the first atom, which could be, for instance,

```
N1
```

This atom (nitrogen, in this example) will be placed at the origin. The second line contains, in order, the atom label for atom 2, the identifier of atom 1, and the distance between atoms 1 and 2. Identifiers can either be atom labels or atom numbers, which correspond to the order the coordinates are listed (1 for the first atom, 5 for the fifth atom listed, and so on); for a second atom in this file, therefore, the identifier for the first atom could be either "N1" or "1". The second atom will be placed along the positive z-axis. For example,

```
N1
C2   N1  1.4589
```

places the carbon atom (C2) at (0.0, 0.0, 1.4589) in Cartesian coordinates. Distances between atoms must be positive.

The third line is made up of five items: the atom label for atom 3, the atom label of one of the previous atoms, the distance between atom 3 and this previous atom, the identifier of the other previous atom, and the angle defined by the three atoms. In this example:

```
N1
C2   N1  1.4589
C3   C2  1.5203   N1  115.32
```

the final line states that atoms C3 and C2 are separated by 1.5203 Å and that the C3–C2–N1 bond angle is 115.32 degrees. The bond angle must be between 0 and 180 degrees, inclusive. The third atom (C3 in this case) is placed in the xz plane (positive x). Clearly, three different atoms must be listed in this line.

Finally, the fourth line contains a third atom identifier of another previously defined atom, and a torsional angle. Consider this example:

```
N1
C2   N1  1.4589
C3   C2  1.5203   N1  115.32
O4   C3  1.2036   C2  126.28   N1  150.0
```

The last line states that atoms O4 and C3 are 1.2036 units apart, that the O4–C3–C2 bond angle is 126.28 degrees, and that the torsional angle defined by O4–C3–C2–N1 is 150.0 degrees. This information is sufficient to uniquely determine a position for O4. If the first three atoms in the torsional angle definition were linear or very nearly linear, O4's position would be poorly defined, however. You should avoid defining torsional angles relative to three colinear (or nearly colinear) angles.

The torsional angle is best understood by "looking down" the "bond" between atoms C3 and C2 (looking from C3 towards C2). In the near field, the C3–O4 bond forms a ray in some direction perpendicular to the C3–C2 axis (ignore the component of C3–O4 along the C3–C2 direction). In the far field, the C2–N1 bond forms another ray perpendicular to the C2–C3 axis. The *magnitude* of the torsional angle is determined to be the angle between these rays (in either direction) chosen such that it is between 0 and 180 degrees. Its sign is positive if this angle is traced clockwise from the ray in the near field to the ray in the far field, and negative if the angle is traced counterclockwise from the ray in the near field to the ray in the far field.

Alternatively, the fourth atom's position can be specified using a second bond angle instead of a torsional angle, if ' 1' or ' –1' is added onto the end of the line. In these cases, the last angle specified

is assumed to be between the first, second, and fourth atoms mentioned on the line (above, the O4–C3–N1 angle). Since there are two possible positions for the atom which meet the angle specifications, the position is defined by the vector product $\mathbf{r}_{12}\cdot(\mathbf{r}_{23} \times \mathbf{r}_{24})$, where $\mathbf{r}_{ij}$ is defined as the vector pointing from the **j**th atom listed on that line to the **i**th atom listed. If this vector product is positive, the value at the end of the line should be ' 1'; otherwise, the value should be ' –1'. You should use torsional angles instead of second bond angles if you want to perform a constrained geometry optimization, however, since Jaguar cannot interpret *any* constraints on bond lengths or angles for geometries containing second bond angles.

All additional lines of the Z-matrix should have the same form as the fourth line. The complete Z-matrix for the example molecule (the 150° conformation of glycine) in Z-matrix form is

```
N1
C2   N1  1.4589
C3   C2  1.5203   N1  115.32
O4   C3  1.2036   C2  126.28   N1  150.0
O5   C3  1.3669   C2  111.39   N1  -31.8
H6   N1  1.0008   C2  113.55   C3  -69.7
H7   N1  1.0004   C2  112.77   C3   57.9
H8   C2  1.0833   N1  108.89   H6  170.0
H9   C2  1.0782   N1  110.41   H6   52.3
H10  O5  0.9656   C3  111.63   C2 -178.2
```

## *Variables and Dummy Atoms in Z-Matrix Input*

Bond lengths or angles can also be specified as variables which are set beneath the Z-matrix itself. This feature makes it easier to input equal values (such as C–H bond lengths or H–C–H bond angles for methane) and also makes it possible to keep several distances or angles the same as each other during an optimization even while they change from their original values.

To use variables, type the variable name (chbond, for instance) where you would normally type the corresponding number (such as a C–H bond length in Å) for each relevant occurrence of that number. You can put a + or – sign immediately before any variable, and you may use several variables if you want to. When you have entered the full Z-matrix, add a line at the bottom setting the variables, such as

```
chbond=1.09  HCHang=109.47
```

At times, the Edit window may list the variable settings on several lines, preceded by a line saying "Z-variables". This format is just another way the interface has of separating the variable settings from the atomic coordinates.

Sometimes, defining dummy atoms can make the assignment of bond lengths and angles easier. Dummy atoms are simply a way of describing a point in space in the format used for an atomic coordinate while avoiding actually placing an atom at that point. Dummy atoms' "elements" are identified as 'X', 'x', or 'Du'. An example of the use of a dummy atom for $CH_3OH$ input follows:

```
C
O    C   1.421
H1   C   1.094    O   107.2
X1   C   1.000    O   129.9    H1  180.0
H2   C   1.094    X1   54.25   H1   90.0
H3   C   1.094    X1   54.25   H1  -90.0
H4   O   0.963    C   108.0    H1  180.0
```

## *Constraining Z-Matrix Bond Lengths or Angles*

You can edit any geometry in Z-matrix format so that the bond lengths or angles you specify are held fixed during a geometry optimization by adding a '#' sign after the appropriate coordinate or coordinates. For example, to fix the HOH bond angle of water to be 106.0 degrees, you could input the following Z-matrix:

```
O
H1  O   0.9428
H1  O   0.9428   H1   106.0#
```

If you performed a geometry optimization on this input geometry, the bond angle would remain frozen at 106˚ throughout the optimization, although the bond lengths would vary. For more details, please see section 4.2, which describes how to set up constraints for optimizations.

If you want to constrain two quantities to be the same as each other during a geometry optimization, use variables in Z-matrix input. See *Variables and Dummy Atoms in Z-Matrix Input* earlier in this section for help. To freeze any variables to remain unchanged during an optimization, add a '#' sign to the end of the variable setting in the line at the end of the geometry input that defines the variables, as in this example, where the C–H bond is frozen at 1.09 Å:

```
chbond=1.09#  HCHang=109.47
```

You should not make any constraint changes from the Edit or Optimization windows while both windows are open, because the Optimization settings could conflict with your hand-assigned constraints.

## *Counterpoise Calculations*

To perform counterpoise calculations, you can input a Cartesian or Z-matrix geometry that includes counterpoise atoms, which have the usual basis functions for that element but include no nuclei or electrons. These calculations can be useful for obtaining an estimate of basis set superposition error (BSSE). For LMP2 calculations (which are described in section 3.2), the LMP2 correction is already designed to avoid basis set superposition error, so we advise computing and adding on only the Hartree-Fock counterpoise correction term.

If you place an at sign ('@') after an atom's label, that atom will be treated as a counterpoise atom. For example, to place sodium basis functions at the Cartesian coordinates (0.0, 0.0, 1.0), you could include the following line in an input file:

```
Na1@   0.0  0.0  1.0
```

Counterpoise atoms can also be included in Z-matrix format geometries by including at signs at the end of atom labels.

## *Specifying Coordinates for Hessian Refinement*

If you are optimizing a molecular structure to obtain a transition state, you may wish to refine the Hessian used for the job. section 4.3 explains the methods used for transition state optimizations, including Hessian refinement; this subsection explains only how to edit your input to specify particular coordinates for Hessian refinement. (Whether or not you refine particular coordinates, you can specify a certain number of the lowest eigenvectors of the Hessian for refinement, as described in section 4.3 in the subsection *Refinement of the Initial Hessian*—the Hessian can be refined in both ways in the same job.)

If you put an asterisk ('*') after a coordinate value, Jaguar will compute the gradient of the energy both at the original geometry and at a geometry for which the asterisk-marked coordinate has been changed slightly, and will use the results to refine the initial Hessian to be used for the optimization. To request refinement of a coordinate whose value is set using a variable, add an asterisk to the end of the variable setting in the line at the end of the geometry input that defines the variables. For instance, if you entered either of the following two input geometries in the Edit window:

```
O1
H2   O1   1.1*
H3   O1   1.1*   H2   108.0*
```

or

```
O1
H2   O1   ohbond
H3   O1   ohbond   H2   108.0*
ohbond = 1.1*
```

they would have the same effect: a job from either input that included Hessian refinement would use both O–H bonds and the H–O–H angle in the refinement.

Molecular symmetry or the use of variables, either of which may constrain several coordinate values to be equal to each other, can reduce the number of coordinates actually used for refinement. For example, for the second water input example shown above, only two coordinates will actually be refined (the O–H bond distance, which is the same for both bonds, and the H–O–H angle); the same would be true for the first example if molecular symmetry is used for the job.

# 2.3. CHARGE AND MULTIPLICITY (STATE)

The State window, whose button appears near the Geometry heading in the main interface window, contains the settings describing the molecular charge and the spin multiplicity of the input molecule. If the molecule you are studying is an anion or cation, you should set the net molecular charge, whose default is 0, by clicking in its box in the State window and editing the value.[2] The spin multiplicity default is singlet, but you can change it to anything up to octet by clicking in the Spin Multiplicity box and making another selection from the option menu.[3] The spin multiplicity is also shown numerically. If the molecular charge and spin multiplicity settings you make do not agree for your molecular input—for instance, if your molecule has an odd number of electrons and you leave the spin multiplicity set to singlet—Jaguar warns you to reset one or the other.

# 2.4. READING FILES

If you already have files containing geometries (either with or without a description of a calculation to perform), you can read them into the Jaguar interface's Read window, which is accessed from the Read button in the top row of the main Jaguar window. The interface can read Jaguar input files generated previously using the Save or Run

2. Keyword **molchg** in **gen** section of input file.
3. Keyword **multip** in **gen** section of input file.

options described in Section 2.7 or files generated from other programs.

This section describes the types of files which can be read in to the interface and explains how to read in these input files. Because files in many format types can be used only to provide geometries (not calculation settings), the first subsection below explains how to read in a geometry from a file without reading any calculation settings from that file. The next subsection explains how for some files (Jaguar input files in particular), you can also read in calculation settings.

## *Reading a Geometry, But No Calculation Settings*

From the Jaguar interface's Read window, you can read a geometry (molecular structure) from a Jaguar input file or from any file of an input type recognized by the program Babel [24]. After reading in the geometry, you can then use the interface to set up Jaguar calculation options for the geometry in question.

When you read in a geometry from a file, Jaguar will also try to obtain information on the molecular charge. Therefore, for any of the file reading techniques described here, the molecular charge may be updated from information in the input file. However, for all non-Jaguar input files, we recommend that you double-check the molecular charge setting in the State window after reading in the geometry.

To read in a file, the interface must know what type of file it is, what directory the file is in, and what the file is called. The default directory searched is the directory where you started the interface, so you may find it easiest to start the interface from the directory containing the relevant input file or files.

When you first select Read, a window appears. This Read window includes information on the current directory whose files are listed, and lists all Jaguar input files in that directory, if any. A Jaguar input file is the standard input for Jaguar jobs, and contains the various settings for the job. Jaguar input files are generated at the beginning and end of runs, or when you use the Save window to save the input without running a job.

If you want to read in the geometry from a file that is not a Jaguar input file, first click on the File Format option menu and select the appropriate file format from the list. Next, set the Read as option menu to Geometry (new job) if you want to make sure all calculation options are set to their defaults (until you change them in other windows of the interface), or set Read as to Initial geometry if you

want to leave intact all calculation settings made from the interface or read into it from previous input files.

If you want to read in the geometry but not the calculation settings from a Jaguar input file, you do not need to alter the setting marked File Format immediately after opening the Read window for the first time, because it is set to Jaguar input by default. However, you must alter the setting marked Read as from its default, Geometry and settings (this default means the interface will read in calculation settings also, as discussed below in the subsection *Reading In Both Geometries and Job Settings*). Change the Read as setting to Geometry (new job) if you want to remove any calculation settings made earlier from the interface or by reading in a previous file's calculation settings. If, instead, you want to leave all calculation settings made from the interface or read into it intact, but change the geometry, change the Read as setting to Initial geometry.

At this point, if the file you wish to read in already appears in the list in the Files box on the right, you can select it by clicking on it. The file you have chosen will then be listed in the Selection bar, and when you click on OK, the interface will read in the geometry from the file.

If the file does not appear in the list, you need to tell the interface where to look for it. The Filter bar at the top of the window indicates what directory and what name type the interface has searched to generate the list of possibilities in the Files box. A '*' is a wild card character, meaning it can represent any text. The word after the last slash is the file name the interface will search for, while the rest of the entry is the directory in which it will search. You can edit the Filter bar by clicking in it and typing, or you can change it by selecting another directory from the Directories box. (If you want to go up one directory, you should select the <current directory>/.. option, where <current directory> indicates the appropriate directory name.)

Once you have altered the Filter bar properly, you can list files meeting that description by clicking on the Filter button at the bottom of the window. As described above, if the file you want to read in shows up in the Files box, simply select it by clicking on its name, then hit OK to read it in.

If you know from the start what directory and file name you wish to select, and do not need to see a list of options in the Files window, you can also bypass the Filter by editing the Selection bar by hand at any point, changing the File Format and Read as settings, if necessary, and hitting OK to read in the specified file.

After you read in a file, you can use other windows to change settings before actually running the job. Most of the rest of this manual concerns calculation options for Jaguar and how to set up Jaguar jobs.

### *Reading In Both Geometries and Job Settings*

The only file types for which Jaguar can read any calculation information *besides* the geometry and molecular charge are Jaguar input files, GAUSSIAN [23] input files, and BIOGRAF [21] Hessian files. The procedure for reading in the geometry and calculation settings from a file of one of these types is exactly like that described above in the subsection *Reading a Geometry, But No Calculation Settings*, except that the Read as option menu in the Read window should be set to Geometry and settings.

When you read the geometry and settings in a Jaguar or GAUSSIAN input file into the interface, the interface updates to show the geometry and all the calculation information in the input file. If you read in a Jaguar input file whose sections contain non-default settings not found in the interface, as described in Chapter 8, all calculation settings will still be read in.

### *Read as Geometry 2 or Geometry 3 Settings*

In the Read window, two options in the Read as option menu, Geometry 2 and Geometry 3, are designed for input only for certain types of transition state optimizations. Therefore, they will be described in section 4.3, which explains special options for transition state optimizations.

## 2.5. GEOMETRY DISPLAY

Several different versions of the Jaguar interface exist in order to support different types of graphics hardware for viewing the molecular structure in the interface's Display window. Jaguar will automatically choose which version of the Jaguar interface to run, depending on the machine type and operating system on your interface host (X client) and display host (X server). The interface executables have the names and uses shown in Table 2.5.1.

The jaguar.mesa interface uses software rendering which is displayed via X11, a process which is significantly slower than the hardware rendering for OpenGL. The advantage of jaguar.mesa is that it works on most X terminals or workstations. If you use an 8-bit color console

***Table 2.5.1.*** *Jaguar interface types.*

| Interface version | Graphics | Supported display hosts/operating systems |
|---|---|---|
| jaguar.opengl | OpenGL | SGI/IRIX 6.5 or later; DEC OSF 1.3 or later |
| jaguar.mesa | X11 | any SGI, IBM, HP, DEC, or PC supported by Jaguar |
| jaguar.none | none | any IBM, HP, or DEC supported by Jaguar |

with this version, there may be colormap flashing as your input focus switches to or from the Display window. The flashing results because private colormap must be used for the rendering to insure a reasonable display. When flashing occurs, some of the buttons on the interface and possibly some of the atom labels may be difficult to read.

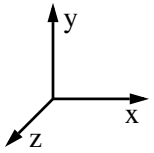
If your display supports an 8-bit TrueColor visual (but not 24-bit), the graphics may look better if you use an 8-bit color visual instead. (The available visuals may be listed using the xdpyinfo command.) In this case, try the command

```
setenv MESA_RGB_VISUAL "Pseudocolor 8"
```

from a display host window to see if it improves the image.

## *Display Window Basics: Orientation & Mouse Control*

When you start the interface, or when you select the Display button near the Geometry heading, a display window will appear. You may need to resize the window using the mouse. If you have provided a molecular structure, a graphical display of it should be visible.




For the Display window, the positive z axis points out, the x axis runs from left (negative) to right (positive) in the window plane, and the y axis runs from the bottom (negative) to the top (positive) of the window.

The display is controlled by the mouse. Note that changing the view of the molecule shown on the display does *not* affect the geometry used by the rest of Jaguar. The mouse commands and their effects on the graphical display are shown in Table 2.5.2.

When you translate the molecule in the z direction, the display actually shows only a slice of the molecule containing a certain range of z values. This option may be useful if, for instance, you want to look at a set of atoms at the "back" of a molecule—by translating the molecule in the z direction, you could make the atoms at the front invisible

***Table 2.5.2.*** *Mouse functionality in the interface's Display window.*

| **If you hold down this button:** | **and then press this key:** | **and move the mouse this direction:** | **you perform the following actions in the display:** |
|---|---|---|---|
| right | (none) | up or down | rotate molecule about x axis |
| right | (none) | left or right | rotate molecule about y axis |
| right | Shift | left or right | rotate molecule about z axis |
| right | Ctrl | right or left | zoom in or out along z axis |
| middle | (none) | any | move molecule in xy plane |
| middle | Shift | left or right | move molecule in z direction |
| middle | Ctrl | left or right | narrow or widen the range of z values visible |

to the display. Similarly, the last option in the table allows you to vary the thickness in the z direction of the layer of molecules visible.

If you hold down the Ctrl key *before* pressing a mouse button, you may end up accidentally resizing the window, depending on your window manager. If this problem occurs, try to make sure to press the mouse button first when entering the commands described.

## *Display Styles*

You can display the molecule in several different styles by choosing the appropriate option under the Display pull-down menu in the Display window: Ball & Stick, GVB Pair, Lines, or Space-filling (spheres only). For the Display options involving spheres (that is, all of them except the Lines option), the atomic radii are chosen according to the Spheres selection under the Options pull-down menu, as described below.

The "sticks" in the ball and stick display are generally from the Jaguar interface's simple calculation of connectivity, where all "bonds" found are displayed as solid cylinders. If, however, the input file is a BIOGRAF .bgf or MacroModel .dat file that contains bonding information, the interface will use this information in the ball and stick display, representing single bonds as solid cylinders, double bonds as two-piece cylinders (with one thin band in the middle), and triple bonds as three-piece cylinders (with two thin bands).

For the GVB pair display, single GVB pairs are displayed as cylinders and multiple pairs as divided cylinders. Lone pairs are currently not displayed. Atoms which are close enough to be considered connected,

yet are not GVB pairs, are connected by thin lines called bond lines. This display option can be helpful if you set GVB pairs from within the interface, as described in section 3.3. As pairs are added from the GVB window, they appear in the display.

## *Labels*

You can choose to have atom labels appear in the display by making a selection other than none from the Display window's Labels pull-down menu. Counterpoise atoms are shown with an '@' symbol, as they are listed in the geometry. If you choose number, the atoms are labeled with the numbers representing the order they appear in the molecular geometry input, while element shows their periodic table labels. To see both at once, you can select element + number. To see the labels you provided in the geometry input, you can choose user label.

## *Other Display Options*

You may make additional adjustments to the display under the Display window's Options pull-down menu. First, you can set the size of the spheres used by several of the Display options.

The Resolution window lets you determine the appearance and smoothness of the spheres with the Sphere depth slide bar. This window also lets you determine the number of sides on any cylinder used to show bonds, with the Cylinder sides slide bar, and the quality of the display shown as you rotate the molecule with the mouse, which is set in the Rotation mode option menu. Generally, the higher the display quality, the more slowly the display will respond to your commands; the connectivity or quick render rotation mode options often lead to much faster rotations than the standard render option.

The Colors window under the display window's Options pull-down menu lets you set the lighting effect on the spheres, the Specular highlights, on or off. You can set the Atom label option menu to show atom labels in colors which are complementary to the color of the atoms they appear on, the same colors as the atoms they appear on, or in the text color, which is set later. The Bond line option menu lets you pick whether the bond lines, such as those appearing in connectivity displays, are portrayed using the atom colors of each atom in the bond or the line color set later. The Text slide bars allow you to determine the relative contributions of red, green, and blue to the text color mentioned above. If you set all three slide bars to .00, you will get black text; if you set them all to 1.00, you will get white text. Similarly, the line color, which appears in the bands on the cylinders repre-

senting multiple bonds and possibly in bond lines as well (see above), can be set with the Line slide bars. The color of the window background appearing behind the molecule can be set with the Background slide bars.

The final selections found under the Options pull-down menu in the display are Realign Geometry and Recenter Geometry. Realign Geometry reverses any rotations and returns the molecular coordinates to their original values, the coordinates used by the rest of Jaguar. Recenter Geometry moves the input structure back to the middle of the window, but leaves any rotations you have made intact.

### *Choosing a Structure To Display*

As will be described later in section 4.3, for certain kinds of transition state optimization jobs, you need to input more than one structure. The Structure pull-down menu in the Display window lets you select which such structure is displayed. The structure will also be identified in the upper-left corner of the Display window.

### *Closing the Display Window*

The Close option, found in the File pull-down menu, allows you to exit the display window, saving any settings you have made there for any future displays of that geometry input, but leaving the molecular coordinates used by the rest of Jaguar intact.

## 2.6. CLEANING UP MOLECULAR GEOMETRIES

### *The Cleanup Button*

When you click the "Clean up geometry" button on the Jaguar Cleanup panel, Jaguar first performs a quick charge-equilibration (Qeq) calculation to obtain partial charges for all atoms in the system, and then uses those charges in an energy minimization which is based on Goddard and Rappe's Universal Force Field (UFF). Because UFF includes parameters for all elements in the periodic table, it can be used for inorganic complexes as well as organic compounds.

During the UFF minimization, the label on the "Clean up geometry" button changes to "Halt cleanup." You can click this button at any time to cut short the minimization. After the cleanup is finished,

Jaguar will reanalyze the symmetry of the molecule and display the point group of the minimized structure. If you're satisfied with the results of the cleanup procedure, you can click the "OK" button to accept the final geometry.

Clicking the "Cancel" button will throw away the cleaned-up geometry and revert to the geometry you had before opening the Cleanup panel.

The convergence criteria for the cleanup minimization are deliberately set fairly loose, so that even large systems can be optimized interactively. In addition, a time limit is imposed on the minimization to keep it from running excessively long. As a result, you may find that the geometry continues to change if you perform a second cleanup minimization on a "cleaned up" structure.

UFF cleanup minimization is useful for quickly bringing a distorted molecule back into the neighborhood of the ab initio minimum-energy geometry, in preparation for full ab initio geometry optimization. However, it is no substitute for ab initio optimization because UFF is a relatively simple force field. It is probably a good idea to perform a cleanup minimization after creating a new molecule with the builder. On the other hand, performing a cleanup minimization on molecule that has already undergone ab initio minimization is likely to move the molecule away from the ab initio minimum. Also, you should be careful to avoid "cleaning up" a structure that has been prepared as an initial guess for a transition-state search.

## *The Symmetrize Molecule Button*

By default, Jaguar takes advantage of molecular symmetry[4] whenever possible, in order to obtain CPU savings. Both Abelian and non-Abelian point groups are recognized (a particular strength of Jaguar).

If you wish, you can turn the use of symmetry off[5] by using the Symmetry pop-up menu in the Methods window of the interface. For some calculations, including GVB, LMP2, GVB-LMP2, and GVB-RCI calculations and calculations of IR intensities or hyperpolarizabilities, symmetry is not yet implemented and will be disabled automatically for the job.

Generally, you should symmetrize the geometry if you plan on using symmetry in the calculation itself. Otherwise, the input coordinates

4. Keyword **isymm** = 8 in **gen** section of input file.
5. Keyword **isymm** = 0 in **gen** section of input file.

may not be accurate enough for all possible symmetry to be recognized.

The Symmetrize window allows you to apply a symmetrization procedure to an input geometry in order to assure that calculations take full advantage of molecular symmetry. You can access the Symmetrize window by clicking on the Symm. button in the main interface window near the Geometry heading.

Since an input geometry must have a certain orientation and sufficient precision in the coordinates before it can be symmetrized, the coordinates must be changed in several ways to generate a new, symmetrized geometry. First, the molecule is translated so that its center of mass is located at the origin of the coordinate system. Second, it is rotated so that it is aligned along the symmetry axes. Third, minor adjustments are made to the coordinates so that the molecule more precisely conforms to the point group.

Note that if you are comparing calculations from geometries which differ only slightly, you must use caution when symmetrizing coordinates. For example, a small symmetry-breaking change can be removed if its magnitude is smaller than the tolerance you have set, which establishes what changes are acceptable. In this case, you should inspect the symmetrized coordinates in the Edit window to insure that symmetrizing had the desired effect and did not wipe out any important information about the molecular geometry.

## *Tolerance*

The tolerance is, roughly, the maximum distance any atom is allowed to be moved while the program searches for the highest possible symmetry for the molecule. Its units are the same as those of the input geometry. A large tolerance yields the highest symmetry but may cause the coordinates to be changed significantly. A small tolerance may yield a lower symmetry, since the coordinates cannot be moved as much. The main Jaguar programs use a small tolerance ($1.0 \times 10^{-6}$ bohr), which should result in molecular energy changes of 1 micro-Hartree or less. You may wish to use a higher tolerance than that when symmetrizing the geometry.

## *Finding the Point Group*

Selecting Find point group will cause the program to find the molecule's point group for the tolerance indicated and display it near the heading Point Group. The geometry does not actually change until you select Symmetrize at the bottom of the window, as described

below, so you can experiment with different tolerances to see what symmetry would be applied for each, if you are careful not to hit Symmetrize

### *Symmetrizing Coordinates*

Selecting Symmetrize will generally change the molecular geometry to a more symmetrical or differently oriented geometry. If you changed it by mistake, you will need to reenter the geometry as described in Section 2.2 and Section 2.4. If you are satisfied that the point group shown is the one you wish to apply to the molecule, you should select Symmetrize. If you are not sure, but wish to apply the point group, examine the resulting coordinates, and then decide whether to use them, you should make sure the original geometry is stored in a file so you can read it in again if necessary.

Select Cancel if you do not wish to change the geometry—for example, if you only wanted to know the symmetry within a certain tolerance, without actually changing the coordinates. Note that in this case the main programs may assume the geometry is of a lower symmetry than the one you saw in the Symmetrize window, and therefore may not take full advantage of molecular symmetry to speed up the calculation.

If you hit Symmetrize, the comment for the job, which is described in Section 2.7 and which appears in the input and output files for the job, will include the note, "Geometry symmetrized to point group," followed by the point group name.

## 2.7. RUNNING JOBS AND SAVING INPUT

You can submit a job either from within the interface or from a command-line prompt. Starting jobs from the interface is easier, but sometimes you might want to save the input files and submit the jobs by hand, in order to use batch queues, submit jobs remotely from a non-X terminal, use scripts for running multiple jobs, or hand-modify the input files. Information on submitting jobs by hand with the "jaguar run" command can be found in Chapter 8, particularly in section 8.1.

### *Starting Individual Jobs from the Interface*

Once you have read in a geometry, you may start a Jaguar job from the interface by selecting Run from the Jobs row of buttons in the

main interface window and entering the appropriate information. When you open the Run window, you should close any other open windows in order to save settings you may have changed.

The information you enter in the Run window is mainly used to tell Jaguar how and where to launch a job. The choices available in the Run window option depend on the jaguar.hosts configuration file. See section 6.1 for more information on this file. If you do not change the entries in the Run window, the settings shown will be used for the run.

If Jaguar is installed on more than one host at your location, you can select which host the calculation will actually be performed on from the option menu next to the Job Host heading. To make a non-default selection, click on the bar and then on one of the choices which appears.

A temporary directory on the calculation host is used to store intermediate files during the calculation. If there is more than one possible choice listed for the Temp directory setting, you should pick one. A subdirectory with the given job name ("h2o", for example) is created within the temporary directory, and the files from the calculation will be stored within this subdirectory. Note that the subdirectory and directory must have sufficient disk space for the job, or it will die.

If you are unsure about whether your temporary directory already exists, you should probably look for it from a terminal window. If it does not exist, you should create it or choose a directory which already exists. If none of the temporary directory choices already exist and you do not wish to create the necessary directories, you can change the jaguar.hosts file so that the interface will offer you different choices (see section 6.1).

The directory listed next to the heading Job directory is the local directory on the interface host where input and output files created by Jaguar will be written. The default local job directory is the directory from which you read the input file, if you read one; otherwise, the default is the directory where you started the interface. If you wish, you may change the default selection by clicking in the box and editing the text there.

If the job host is identified in the jaguar.hosts file as having more than one processor, that information will be indicated in the box marked # of Processors. If this number is greater than one, the job will run in parallel.

The text in the box headed Job name determines the names of many of the files created by Jaguar, as well as the name of the subdirectory

within the temporary directory, which is described above. The files whose names depend on the job name include the input file, the log file (which shows the job's progress), and the output file listing the calculation results. For instance, if the job name is "h2o", the results are stored in a file called "h2o.out" within the local job directory.

The default setting for the job name is the base of the input file name, if any, from which the molecular geometry was read. For example, if you told the interface to read the geometry from a file called "h2o1.in", the default job name setting would be "h2o1". You can change the job name by editing it from the Job name bar in the Run window or from the Job Name box in the main interface window, by clicking on the bar and typing in a name. If you did not read in the geometry from a file, you should enter a job name in either the Run window or the main interface window.

Any text entered in the box marked Comment will appear in the input and output files for the job. If you symmetrize the geometry, a procedure described in Section , the comment will contain text noting that the geometry was symmetrized to a certain point group. You can enter other text describing the job for your own convenience. *The comment should not contain any $ or & characters.* The comment appears in the input file immediately before any keyword settings corresponding to later interface selections, and in the output file under the heading "Input file comments".

By default, all temporary files and directories are deleted when the job finishes, after the output file, restart file (which is described in section 6.4), and other useful files are copied back to the local job directory. If you want to save the binary files generated in the temporary directory's job subdirectory and used during the run, you may do so by selecting that option from the menu in the bar next to the heading Scratch Files. Note, however, that these files are often large and should only be saved if necessary, and that any files in the temp directory may be deleted automatically if your site has "scratching."

When you are satisfied with the run-time settings, you can start the job by clicking on the RUN button at the bottom of the window. You can then check the current status of the job from the Job Status window, as described in Section 2.8. If you submit additional jobs, they will run concurrently. If you exit the Jaguar interface, any Jaguar jobs still running will continue to run to completion.

## *Running Batch Jobs or Scripts from the Interface*

You can run multiple Jaguar jobs sequentially using the Batch window of the Jaguar interface, which is accessible from the Batch button near the Jobs heading in the main interface window. This window lets you launch many jobs at once on the machine upon which you started the interface, changing calculation options in many input files if you wish. For instance, you can

- run a series of Jaguar jobs without changing the input files, launching each job only after the preceding job has completed;
- run the same type of job for several input geometries, launching each job only after the preceding job has completed; and
- run series of jobs in which later jobs use input files generated during earlier jobs, which include the results from the earlier jobs.

Several Jaguar batch scripts are included with the program. You can also write your own Jaguar batch scripts. section 8.20 explains what you can do with Jaguar batch scripts and what sort of format they should have. If you write your own batch scripts, make sure their file names end in the suffix ".bat".

To run a Jaguar batch job from the Batch window, you first need to select a batch script. The batch script can be in any of three directories:

- the batch script directory installed with Jaguar (identified in the interface as BUILTIN_SCRIPTS), whose location is hardwired in the Jaguar installation directories;
- your own personal Jaguar batch script directory, which is set by the environment variable JAGUAR_SCRIPTS if you have set it, and is otherwise assumed to be ~*yourname*/jaguar_scripts, where *yourname* is your user name; or
- the current directory, which is the directory containing the last input file you read in or wrote out or, if you have not read or written any files from the interface, the directory where you started the interface.

To select a Jaguar batch file in any of these directories, first click on the Select button in the upper right corner of the Batch window. The Select Batch Script window will open. By default, this window will

show that the script selected is the JOBS.bat script. The Notes window shows comments from the JOBS.bat script; as they indicate, this script simply runs a series of jobs from the input files it is passed.

To select another one of the built-in scripts included with Jaguar, just click on its name in the list labeled Scripts, then click OK. To select a script from your own Jaguar batch script directory or from the current directory (both defined above), change the option menu in the middle of the window to the setting User scripts or Local scripts (respectively), click on the name of the script you want to select, then click OK.

Jaguar batch scripts can either require that you specify an input file (or list of input files) to be run with the "recipe" in the batch file, or they can include a self-contained list of Jaguar input files. If the script you have selected processes input files, you next need to select an input file (or list of files) to be fed to the batch script after you have selected the batch script. (If you are not sure whether the script processes input files or is a self-contained script, reopen the Select Batch Script window and look at the text in the middle of the window.)

Select input files from the window marked Inputs, which will list Jaguar input files in the directory shown in the Path box. The input files will be passed to the batch script in the order in which they appear in the list. You can select the first input file by clicking on its name. You can add other input files to the list by pressing the Shift key on your keyboard and selecting another file, which will add the whole range of files between the two to the list, or by pressing the Control key on your keyboard, then clicking on each additional file name you want to select. As you select files, their names will be highlighted. You can also use the (Un)Select All button at the bottom of the window to select or unselect all the files in the list, or use the Current job button to select the input you last read in or edited from other interface windows.

By default, the restart files produced at the end of Jaguar jobs, which are named in the form jobname_*xx*.in, where *xx* is a two-digit number, will not be listed in the Inputs file list. If you want to list them, click the box marked Hide restart files to deselect it. You can turn this option back on by clicking on it again.

After you have finished selecting the Jaguar batch script and (if necessary) Jaguar input files, click the RUN button to launch the batch job. The Jaguar jobs run as part of this batch job will run sequen-

tially—that is, no job will start running until the previous job in the list (if any) has finished running.

Immediately after you click RUN, the Job Status window will open. This window will show you the batch log file(s) (*.blog) for the batch job. The information will be automatically updated as the Jaguar jobs run. If you close this window, you can reopen it by clicking on the Check button near the Jobs heading in the main interface window.

## *Saving Input Files*

You can use the Save window to store a Jaguar input file generated by the interface, or to save a geometry in an appropriate format for another program. You can later scan Jaguar input files back into the interface, as described in Section 2.4, and run jobs from the interface, as described earlier in this section. Alternatively, you can use a Jaguar input file as input for a job submitted from the command line—that is, not using the interface. You would need to start jobs by hand if you wanted to use batch queues or submit jobs remotely from a non-X terminal. Information on submitting jobs by hand can be found in Chapter 8, especially in section 8.1. Jaguar input files can be copied to other machines with Jaguar and used for runs there.

In the Save window, which you can access by clicking the Save button in the top row of the main interface window, the directory listed next to the heading Input file directory is the directory on the interface host where the file saved by Jaguar will be written. The default input file directory is the directory from which you most recently read a file into the interface, if you read one; otherwise, the default is the directory where you started the interface. If you wish, you may change the default selection by clicking in the box and editing the text there.

The text in the box headed Job name determines the name of the input file created by Jaguar. For instance, if the job name is "h2o" and you save a Jaguar input file, the input file saved will be called "h2o.in". The default setting for the job name is the base of the input file name, if any, from which the molecular geometry was read. You can change the job name by editing it from the Job name bar in the Save window or from the Job Name box in the main interface window, by clicking on the bar and typing in a name. If you did not read in the geometry from a file, you should enter a job name in either the Save window or the main interface window.

You can save files in a variety of formats for other programs by making the appropriate selection from the Save as menu bar. For any

file formats other than the Jaguar input (*.in) file, only the geometry will be included in the file. The file's name will be determined by appending the extension indicated in the file type list to the job name.

Any text entered in the box marked Comment will appear in the input file for the job. If you symmetrize the geometry, a procedure described in Section , the comment will contain text noting that the geometry was symmetrized to a certain point group. You can enter other text describing the job for your own convenience. *The comment should not contain any $ or & characters.* The comment will appear in the Jaguar input file immediately before any keyword settings corresponding to later interface selections.

### *Output*

A Jaguar log file contains comments on the progress of a job. If the job was started from the interface, the log file is written to the local job directory selected in the Run window. The log file notes when each section of Jaguar is complete, as well as noting data from each iteration in an SCF calculation as it is calculated. You can look at this file by using the Job Status file viewer window, which comes up when a job is launched or when you click the Check button from the main window. When a job is running, the log file is displayed in the Job Status window. See section 5.8 for more information on this file.

The primary Jaguar output is contained in the output file, which is initially created on the temporary directory of the host where the calculation is performed, but is also copied back to the interface host when the job is complete. The output file is described in Chapter 5. From the file viewer window, once the log file shown for a job indicates that the job has completed, you can look at the corresponding output file by clicking the View File button at the top of the window, selecting the appropriate output file from the resulting list by clicking on it, and hitting OK.

## 2.8. OTHER INTERFACE FEATURES

Some other features of the interface which are not covered elsewhere are briefly described here. Note that sometimes a menu item is dimmed (meaning that its name appears in a different, usually less intense color), which means that this option is currently unavailable. For example, the Run button is dimmed until a geometry is entered.

## *Checking Jobs with the Job Status Window*

The Job Status window allows you to examine Jaguar log files, output files, or any other text files. It opens automatically when you start a job. If you close the Job Status window, you can reopen it again later by clicking the Check button from the main window.

The log file for the last job you run (h2o.log for the job "h2o", for example) will be displayed automatically in the Job Status window. The log file indicates when each Jaguar program has finished running. section 5.8 contains more information about this file.

The View File button allows you to display the output file for the job, or any other text file, in the Job Status window. This window allows you to specify which file you want to display, in much the same way as you pick a file name when you want to read the geometry from a file. (See Section 2.4 for more information.) Note that by default, only output files are listed, but you can alter this search pattern by editing the Filter bar and hitting the Filter button at the bottom of the window.

Once you have chosen to display a file, its path and name will appear as an option when you select View File. You can use the New File option to display other files. Up to ten files can be listed under the View File option menu.

## *Resetting and Quitting*

The Reset option, which near the Jobs heading in the main interface window, clears many of the settings, returning them to the defaults. Selecting Reset wipes out the geometry and any other files read in, as well as all settings describing the wavefunction and properties to be calculated and any settings you may have made with the Geometry or Output buttons. It also sets the Job name value in the Run or Save window to whatever is appropriate when you input the next geometry. However, the other selections you have made in the Run or Save window will remain the same. Reset usually prompts you to make sure you are willing to wipe out any changes. This question, like all those asked in Warning windows, must be answered before the interface will let you do anything else.

Selecting Quit closes the interface. If you have started a job, it will continue to run unless you kill it from your terminal window.

## *About and Help Buttons*

The About button displays information about Jaguar and Schrödinger. You must close it before using other parts of the interface.

The Help button in the upper right corner of the main interface window brings up the Help window. You can see on-line help on a variety of subjects by clicking on them as they are listed under the Help items heading so that they show up in the Selection bar, then hitting Select. You can also obtain help from the windows you use by pressing the Help buttons found within them. Pressing Help then brings up the Help window with the appropriate topic already selected. All of the information in the on-line help is also included in this manual.

## *Editing Input*

The Edit Input button near the Job Name box in the main window of the interface brings up the Edit Job window, which allows you to make settings using the input file keywords for the **gen** section of the Jaguar input file, which are described in section 8.6, or alter any other input file settings. Formatting cannot be altered from the Edit Job window, however.

The options available for editing jobs from the Edit Job window appear under the pull-down menus File and Edit. These options function exactly like the corresponding options in the Edit window for editing geometries. See *Inputting or Editing a Geometry Within the Interface* in Section 2.2 for more details. Changes you make from the Edit Job window will not be saved outside of the interface until you select OK in the Run or Save window.

You do not need to use the Edit Job window to do anything described in Chapters 2 through 5; you can use the rest of the interface to set up calculations in all the ways described in these chapters. However, if you prefer to set up jobs with keywords, or if you want to use any options described in Chapter 8 that are not included in the interface, the Edit Job window provides you with a convenient way to do so. If you would like to use keyword settings instead of interface windows to set options that appear in the interface, please refer to the footnotes in Chapters 3 and 5 to find out which keyword settings are appropriate. If you make and save a setting in the Edit Job window that corresponds to something shown in an interface window, the interface selection will show the change.

From the Edit Job window, if you select a keyword with the mouse by double-clicking on it or dragging over the keyword, the keyword will often be described briefly at the bottom of the window. To see a fuller description of the option, you can then access the online help relevant to that option by clicking Help near the top of the Edit Job window.

If the input file contains input the interface does not understand (such as a keyword it does not recognize), you will get a warning that the keyword is unrecognized when you run the job or save the input file. If you hit OK in answer to this the warning, the unrecognized information will be retained in the input for the job.

## *Other Main Window Options*

Most of the scientific settings available are described in the next chapter. The Output buttons, which can be used to request additional information in output files, are described in Chapter 5.

# 3. Options

You can make many of the calculation settings for Jaguar jobs using the windows accessed by clicking on the following buttons in the main interface window:

| | |
|---|---|
| **DFT** | density functional theory calculations |
| **Local MP2** | local Møller-Plesset second-order perturbation theory calculations |
| **GVB** | generalized valence bond calculations |
| **Optimization** | geometry optimization and forces |
| **Solvation** | solvation energy calculations |
| **Properties** | multipole moments and charge fitting and hyperpolarizability properties |
| **Frequencies** | vibrational frequencies, IR intensities, and thermochemical properties |
| **Basis Set** | basis set options |
| **Methods** | initial guess, convergence, orbital localization, and various technical settings |

Optimizations are described in Chapter 4, and the other interface options are described in Chapters 2 and 5.

Throughout this chapter, footnotes indicate the Jaguar input file keywords and sections that correspond to particular interface settings. If you are working from the interface, you can ignore these footnotes, but you may find them helpful if you decide to use input files to submit jobs without using the interface or if you want to edit

keywords directly by using the Edit Job window described in Section 2.8 under *Editing Input*.

# 3.1. DENSITY FUNCTIONAL THEORY (DFT) SETTINGS

Jaguar's density functional theory module allows you to employ any of a variety of functionals to describe exchange-correlation matrix elements for either open or closed shell systems, while evaluating the Coulomb (and the exact (HF) exchange, if you wish) operator pseudospectrally. The theory itself is described in section 7.5; this section describes how to set up a DFT calculation within Jaguar. You can perform DFT geometry optimizations, solvation calculations, charge fitting, and all other calculations and properties available for Hartree-Fock wavefunctions. You can also specify a functional or set of functionals to use for a non-self-consistent DFT evaluation of the energy of the final, post-SCF HF or GVB wavefunction.

## *Stage and Grid Density*

By default, the Stage setting at the top of the window is set to SCF, meaning that when you request a DFT calculation, Jaguar performs an SCF calculation of the DFT wavefunction and energy.[1] If you set Stage to Post-SCF, however, Jaguar will instead evaluate the energy of the final, post-SCF wavefunction using the functionals you have specified.[2] You may also choose one set of functionals for the SCF stage and another set for a post-SCF DFT energy evaluation by making functional settings for each Stage choice in turn. If you do a post-SCF DFT energy evaluation on any wavefunction, you cannot perform a geometry optimization or calculate the solvation energy, polarizability, or any other non-default properties.

The other menu at the top of the DFT window, the Grid density menu, determines the grid for DFT calculations. By default, DFT calculations use grids with a medium point density,[3] but finer density grids are also available.[4]

1. The keyword **idft** in the **gen** section of the input file determines functionals for the SCF calculation.
2. The keyword **jdft** in the **gen** section of the input file determines post-SCF functionals.
3. Keywords **gdftmed** = –10, **gdftfine** = –11, and **gdftgrad** = –12 in **gen** section of input file.
4. Keywords **gdftmed**, **gdftfine**, and **gdftgrad** = –13 in **gen** section of input file.

The rest of the settings in the DFT window determine the functional or functionals used, if any. Unless you select a functional or functionals, no DFT calculation will be performed.

## *Method Options*

The most commonly used combinations of functionals can be set easily from the buttons under the Method heading, while the settings under the Functionals heading offer somewhat more flexibility. As you make selections under the Method heading by clicking in the appropriate labels, the settings shown under the Functionals heading change accordingly. By default, DFT is not used, so the default Method setting and Functionals settings are all none.

The Method options include both non-hybrid methods, in which the full HF or GVB and DFT terms are included in the SCF or post-SCF calculation, and Becke's Half & Half [25] and three-parameter [26] methods, which include a term for the Hartree-Fock-type treatment of the exchange and weight the contributions of each term with appropriate coefficients. The non-hybrid settings, which are identified as local or non-local density approximation methods (LDA or NLDA), employ the following functionals:

- LDA: VWN.[5] Slater local exchange functional [27], Vosko-Wilk-Nusair (VWN) local correlation functional [28].

- NLDA: BLYP.[6] Slater local exchange functional [27], Becke's 1988 non-local gradient correction to exchange [30], Lee-Yang-Parr non-local correlation functional [31].

- NLDA: BP86.[7] Slater local exchange functional [27], Becke's 1988 non-local gradient correction to exchange [30], Perdew and Zunger's 1981 local correlation functional [32], Perdew's 1986 gradient correction functional [33].

- NLDA: GGA-II.[8] Slater local exchange functional [27], Perdew-Wang 1991 local correlation functional with Perdew-Wang 1991 GGA-II non-local exchange and non-local correlation functionals [29], where "GGA" stands for "generalized gradient approximation."

5. Keyword **idft** = 101 in **gen** section of input file.
6. Keyword **idft** = 2011 in **gen** section of input file.
7. Keyword **idft** = 1311 in **gen** section of input file.
8. Keyword **idft** = 4441 in **gen** section of input file.

The hybrid methods employ either the parameters developed for Becke's three-parameter method [26] (Becke 3) or the parameters developed for Becke's Half & Half method [25]. These parameters determine the contributions of terms from each functional and from the computed Hartree-Fock exchange. The Method options for hybrid methods employ the following terms:

- Becke 3: LYP.[9] Exchange: exact HF, Slater local exchange functional [27], Becke's 1988 non-local gradient correction [30]; correlation: Vosko-Wilk-Nusair (VWN) local functional [28] and Lee-Yang-Parr local and non-local functional [31].

- Becke 3: P86.[10] Exchange: exact HF, Slater local exchange functional [27], Becke's 1988 non-local gradient correction [30]; correlation: Vosko-Wilk-Nusair (VWN) local functional [28] and Perdew's 1986 gradient correction functional [33].

- Becke 3: GGA-II.[11] Exchange: exact HF, Slater local exchange functional [27], Becke's 1988 non-local gradient correction [30]; correlation: Perdew-Wang 1991 local correlation functional and GGA-II non-local correlation functional [29].

- Half & Half.[12] Exact Hartree-Fock exchange and Slater local exchange functional [27]; no functional for correlation.

## *Functionals*

The DFT window's Functionals settings provide information on which functionals are currently selected and allow you to make functional choices not available from the Method options described above.[13] Any time you make a choice from the buttons under Functionals that does not correspond to a Method option, the Method option will change to Custom.

The Hybridization menu's default setting, none, indicates that the full HF or GVB and DFT terms are to be included in the SCF or post-SCF calculation. If the Hybridization choice is Half & Half or Becke 3 par., the Hartree-Fock-type treatment of the exchange and the contributions of the functional terms selected will be weighted by coefficients from either Becke's Half & Half or three-parameter method.

9. Keyword **idft** = 22111 in **gen** section of input file.
10. Keyword **idft** = 21111 in **gen** section of input file.
11. Keyword **idft** = 24411 in **gen** section of input file.
12. Keyword **idft** = 10001 in **gen** section of input file.
13. DFT keyword settings are extensive and complicated, so further options are not footnoted. See the *DFT Keywords* subsection of section 8.6 for more information.

Once you have specified the hybridization type (if any), you can choose the exchange functional or functionals. The only local exchange functional available from the interface is Slater's [27], which can be used by itself or with either the Becke 1988 [30] or the Perdew-Wang 1991 GGA-II [29] non-local exchange correction term. If you select none for the exchange functional, Jaguar will use the exact Hartree-Fock exchange; however, if you are using a hybrid method, you should always select some sort of exchange functional.

The other Functionals settings determine the functionals used to treat correlation, if any. The local correlation functional options available within Jaguar are two functionals by Vosko, Wilk, and Nusair [28] (labeled VWN and VWN5), Perdew and Zunger's 1981 functional [32] (labeled PZ81), and Perdew and Wang's 1991 local correlation functional [29]. Non-local (NL) correlation options are Perdew's 1986 gradient correction functional [33] (labeled PW86), Perdew and Wang's 1991 generalized gradient approximation correlation functional [29] (labeled GGA-II), or the Lee-Yang-Parr exchange functional [31], which includes both local and non-local terms (as indicated in its label, LYP (L+NL)).

As an example, to use the Functionals settings to set up a calculation using Becke's three-parameter method to weight the Slater/Becke 88 exchange functional, Perdew and Zunger's 1981 local correlation functional, and Perdew's 1986 non-local gradient correction functional, you could select Becke 3 par. from the Hybridization menu, Slater/Becke 88 from the Exchange menu, PZ81 from the Local Correlation menu, and Perdew 86 from the NL Correlation menu. As another example, you could select the Half & Half option from the Method choices, then add functionals by selecting, for instance, Slater/Becke 88 to add Becke's non-local gradient correction to the exchange functional, and LYP (L+NL) for a correlation treatment.

As usual, any time you select the OK button at the bottom of the window, the settings shown will be preserved until they are both changed and saved. If you hit Cancel, none of the settings made since you opened the window will be saved.

# 3.2. LOCAL MP2 SETTINGS

The Local MP2 button brings up a window that allows you to set up a local Møller-Plesset second-order perturbation theory [34, 35, 36, 37] calculation. The local MP2 (LMP2) method greatly reduces the basis set superposition errors that can arise from the canonical MP2 method [37]. The LMP2 method is much faster than canonical MP2 and typi-

cally recovers 98% of the canonical MP2 energy correction. The pseudospectral implementation of LMP2 is described in section 7.4.

## *Summary of the LMP2 Method in Jaguar*

For closed shell systems, you can perform LMP2 geometry optimizations, charge fitting, solvation calculations, and most other options available with HF wavefunctions. Local MP2 geometry optimizations employ analytic gradients. For calculations of LMP2 dipole moments, Jaguar computes a coupled perturbed Hartree-Fock (CPHF) term, which can be computationally expensive. However, since CPHF methods lead to a better description of the charge density, we recommend computing LMP2 dipole moments as well for any calculation for which you need to compute accurate LMP2 electrostatic potential (ESP) fitted charges. (For more details, see the subsections *Electrostatic Potential Fitting* and *Multipole Moments* in Section 3.6.)

Currently, Jaguar's implementation of the local MP2 method requires basis sets that allow the pseudospectral method to be used. This basis set information can be found in Section 3.8 and in several of the periodic tables of information by element in the back of the User's Guide. The interface also warns you if Jaguar will attempt a non-pseudospectral, all-analytic calculation.

The local MP2 reference wavefunction is produced through Pipek-Mezey localization [39] of the usual Hartree-Fock reference wavefunction, a procedure which involves a unitary transformation of the occupied canonical Hartree-Fock orbitals. This localization procedure, which is similar to the localization of bond pairs for the GVB representation of a molecule, *does not* change the wavefunction's energy from the usual canonical reference wavefunction's.

In LMP2, unlike in canonical MP2, each occupied orbital's space of correlating virtual orbitals is limited to those orbitals which are localized on the atoms of the local occupied Hartree-Fock orbital. The localization of the occupied orbitals makes this limitation of the virtual space a good approximation, and allows the reduction of basis set superposition errors which arises from this method. In the limit that all local virtual orbitals are assigned to every occupied orbital, the local MP2 method and the canonical MP2 method are exactly equivalent.

All calculation types available for LMP2 wavefunctions are also available with the "local local" MP2 method, which allows you to treat only some atoms at the LMP2 level, while the remaining atoms

are treated at the HF level. Local local MP2 calculations use the Pipek-Mezey localized orbitals that are localized on the specified atom pairs. The atomic orbital coefficients for each Pipek-Mezey orbital are evaluated and summed for each atom, and the orbital is considered localized on the two atoms whose coefficient sums are largest. If the largest coefficient sum on one atom is more than ten times as large as the coefficient sum on any other atom, the Pipek-Mezey orbital is considered to be localized on that single atom, and that Pipek-Mezey orbital will be included in any LMP2 calculation for which that atom is specified in any requested LMP2 atom pairs.

## *Setting Up an LMP2 Calculation*

Jaguar will not perform an LMP2 calculation unless you make a selection under Correlated Pairs to indicate which atoms should be treated at the LMP2 level. To perform an LMP2 calculation that includes all atoms, select LMP2 (all pairs).[14]

Alternatively, you can perform a "local local" MP2 calculation by specifying that only some atoms should be treated with LMP2, while the remaining atoms are treated at the HF level. Jaguar includes a setting to automatically treat all atoms bonded to atoms of other elements (except C atoms bonded only to C and H atoms) at the LMP2 level. (We recommend this setting for solvation calculations using LMP2.) To request such a calculation, select LMP2 (hetero pairs).[15]

If you want to specify atom pairs yourself, select LMP2 (user-input pairs) and input the LMP2 pairs in any order using the three slide bars in the LMP2 window.[16] To specify the first LMP2 pair, leave the slide bar marked Pair set at 1. Next, specify the atom numbers for the atoms in that pair by sliding the squares in the Atom 1 and Atom 2 slide bars or by clicking in the bar. The appropriate atom labels (for example, "H2") will appear to the right of the Atom 1 and Atom 2 boxes and will reflect the atoms selected in those boxes. Any additional pairs can be entered in the same manner after specifying a new pair number with the Pair slide bar. You can also combine the user-input pairs and heteroatom pairs options, setting your own LMP2 pairs in addition to all heteroatom pairs.

14. Keyword **mp2** = 3 in **gen** section of input file.
15. Keywords **iheter** = 1 and **mp2** = 3 in **gen** section of input file.
16. If you were editing an input file directly instead of using the interface, you would need to set LMP2 pairs in the **lmp2** section of the input file.

# 3.3. GENERALIZED VALENCE BOND (GVB) SETTINGS

The window found by selecting GVB allows you to request a generalized valence bond (GVB) calculation and to set the GVB pairs for that calculation. You can also enable a restricted configuration interaction (RCI) calculation [12, 40, 41] for some or all of the pairs. The theory behind GVB and GVB-RCI calculations is explained in section 7.2 and section 7.3.

The default Jaguar calculation is closed shell or open shell Hartree-Fock, depending upon the number of electrons in the system. If you want to include electron correlation by using the Generalized Valence Bond Perfect-Pairing (GVB-PP) method [20], you can provide a list of GVB pairs to be used in the calculation. This list will automatically enable GVB.

The GVB and GVB-RCI methods in Jaguar do not include the concept of resonance. Consequently, the GVB or GVB-RCI pair input for a molecule such as benzene, for example, should include alternate single and double bonds for its carbon ring. If you perform a GVB or GVB-RCI geometry optimization on a molecule with equal, resonating bonds (like the carbon bonds in benzene), you should force the optimizer to keep their bond distances the same, even if the input lists different bond orders for the bonds. To impose this restriction, use Z-matrix form for your geometry input and set all relevant bonds equal to the same variable. See the subsections *Z-Matrix Format for Geometry Input* and *Variables and Dummy Atoms in Z-Matrix Input* in Section 2.2 if you need help with this procedure.

## *GVB or GVB-RCI Pair Input*

In order to describe the placement of GVB pairs for a GVB calculation,[17] you will need to know the atom numbers for the relevant atoms. If you are able to use the Display option found under the Geometry heading in the main window, you can display the atom numbers and elements, as described in Section 2.5, and use the information to set GVB pairs. The pairs you describe will, in turn, show up on the display, if you have chosen to show GVB pairs there. If you are unable to use the Display option, you can choose the Edit option from under the Geometry heading and identify the atom number by the order the atoms are listed in the file.

17. If you were editing an input file directly instead of using the interface, you would need to set GVB pairs in the **gvb** section of the input file.

You can input GVB pairs in any order. To specify the first GVB pair, leave the slide bar marked Pair set at 1. Next, specify the desired pair type by clicking in the appropriate box. You can choose a sigma bond, pi bond, second pi bond (in a triple bond), or lone pair. If you select a lone pair, it is assigned to a single atom, so when you specify the number of the atom for the lone pair by sliding the squares in the Atom 1 slide bar or by clicking in the bar, the same number is reflected for Atom 2. If you select any other kind of pair, you must set the atom number for Atom 2 separately. The appropriate atom labels (for example, "H2") will appear to the right of the Atom 1 and Atom 2 boxes and will reflect the atoms selected in those boxes.

Additional GVB pairs can be specified in the same way as the first pair, after changing the number shown by the Pair slide bar. *A particular atom should have either all or none of its lone pairs specified as GVB lone pairs.* Also, you cannot set GVB lone pairs when you are using a minimal basis set (e.g. STO-3G).

If you compute solvation energies using GVB or LMP2, as described in Section 3.5, we recommend using heteroatom pairs for the GVB calculation for the most efficient results, particularly since solvation energy calculations often use radii optimized for calculations with heteroatom pairs set. (See section 9.6 for more details.) Heteroatom pairs are all pairs whose atoms are of different elements, except for C–H pairs.

You can select RCI pairs in two different ways. By default, RCI is off for all pairs, and the RCI pair selection bar at the bottom of the GVB window is set to user selected. In this case, a pair will only be included in an RCI calculation if you click the RCI for this pair button beside the Pair slide bar. Alternatively, if you want to set RCI on for all pairs, you can change the setting in the RCI pair selection option menu to RCI on for all pairs. The RCI for this pair button will dim, but will be set automatically on for all pairs.

# 3.4. GVB-LMP2 CALCULATIONS

Jaguar's pseudospectral GVB-LMP2 module allows this multireference perturbation method to be applied to medium and large molecules within reasonable CPU, memory, and disk limits. The method has been shown to give highly accurate conformational energetic results in particular [18].

For GVB-LMP2 calculations, Jaguar first performs an SCF calculation of the reference GVB wavefunction, using the GVB pairs speci-

fied in the input. Next, the program applies an LMP2 perturbative correction to the energy. The LMP2 calculation is performed on the entire system, even if only part of the system was treated at the GVB level.

To set up a GVB-LMP2 calculation, you must first specify the GVB pairs to be used in the GVB reference wavefunction, following the procedure described in Section 3.3. Next, you must request an LMP2 treatment on all pairs in the system, by choosing LMP2 (all pairs) from the Pairs option menu in the Local MP2 window. You can perform the LMP2 calculation at either the valence only or the all electrons level.

We advise using GVB-LMP2 primarily for single-point energy calculations; Jaguar cannot compute GVB-LMP2 atomic charges or analytic gradients. For best results with GVB-LMP2, first run your calculations with the 6-31G** basis set, then change the basis set in the restart file to cc-pVTZ(-f) and restart the job. (See section 6.4 for a description of how to restart jobs.) This procedure will generally be significantly faster than running a GVB-LMP2/cc-pVTZ(-f) job from scratch.

The most effective choice of GVB reference wavefunction depends upon the type of calculation being performed. For conformational energy calculations, we recommend setting all possible GVB lone pairs, all possible GVB pairs that describe multiple bonds between two carbons, and all GVB pairs for bonds between two different non-hydrogen atoms. (For information on how to make these GVB pair settings automatically, please see *GVB and Lewis Dot Structure Keywords* in section 8.6.) For studies of bond dissociation, all bonds from the atoms involved in the dissociating bond (or bonds) should be treated at the GVB-LMP2 level. (Note also that for dissociation of multiple bonds, GVB-LMP2's accuracy is limited by its inadequate treatment of spin coupling between high-spin fragments; we recommend using the GVB-RCI method instead for such cases.)

# 3.5. SOLVATION

Jaguar can treat solvated molecular systems with a self-consistent reaction field method, using its own Poisson-Boltzmann solver [15], [116].[18] You can compute solvation energies and minimum-energy solvated structures or solvated transition states. To perform a geometry optimization in solution, you must make appropriate settings in

18.Keyword **isolv** = 2 in **gen** section of input file.

the Optimization window as well. The solvation energy from a geometry optimization is computed as the difference between the energy of the optimized gas phase structure and the energy of the solvated structure that was optimized in solution. (To avoid optimizing the structure in the gas phase, use the option described below under *Performing or Skipping a Gas Phase Optimization*.)

In the SCRF method used by Jaguar, Jaguar first calculates the usual gas phase wavefunction and from that the electrostatic potential, and fits that potential to a set of atomic charges, as described in *Electrostatic Potential Fitting* in Section 3.6. These charges are passed to the Jaguar Poisson-Boltzmann solver, which then determines the reaction field by numerical solution of the Poisson-Boltzmann equations and represents the solvent as a layer of charges at the molecular surface (which serves as a dielectric continuum boundary). These solvent point charges are returned to Jaguar's SCF program, which performs another quantum mechanical wavefunction calculation, incorporating the solvent charges. This process is repeated until self-consistency is obtained. The cost is roughly twice that of a gas phase calculation.

Solvation energies can be computed for cases using HF, DFT, GVB, or LMP2 wavefunctions. For GVB or local LMP2 solvation energy calculations, we recommend using heteroatom pairs for the most efficient results, particularly since solvation energy calculations often use radii optimized for calculations with heteroatom pairs set. (See section 9.6 for more details.) Heteroatom pairs are all pairs whose atoms are of different elements, except for C–H pairs. See Section 3.2 and Section 3.3 for information on setting LMP2 or GVB pairs.

## *Solvent Parameters*

If the solvent you wish to use for your solvation energy calculation is listed in the set of choices available under the Solvent option menu, you can make the appropriate choice, and Jaguar will automatically perform a solvation calculation, setting the appropriate dielectric constant[19] and probe radius.[20] The dielectric constant [44] and probe radius [45] values set by Jaguar for various solvents are shown in Table 3.5.1.

Alternatively, if you want to use a solvent that is not on the list, you can define it after choosing other from the Solvent option menu. To define your solvent, change the entries for Dielectric constant, Solvent molecular weight, and Solvent density. The latter two values are used to calculate the probe radius (in Angstroms), whose value is shown in

19.Keyword **epsout** in **gen** section of input file.
20.Keyword **radprb** in **gen** section of input file.

***Table 3.5.1.*** *Parameters for various solvents.*

| Solvent | dielectric constant | probe radius |
|---|---|---|
| cyclohexane | 2.023 | 2.78 |
| carbon tetrachloride | 2.238 | 2.67 |
| benzene | 2.284 | 2.60 |
| chlorobenzene | 5.708 | 2.72 |
| 1,2-dichloroethane | 10.65 | 2.51 |
| methanol | 33.62 | 2.00 |
| nitrobenzene | 35.74 | 2.73 |
| water | 80.37 | 1.40 |

the same window, according to the procedure defined in note [45] in the back of the User's Guide.

### *Performing or Skipping a Gas Phase Optimization*

If you are computing the solvation energy of a minimum-energy or transition state structure optimized in solution, your calculation should compare the energy of the optimized solvated structure to the energy of the optimized gas phase structure. Therefore, by default, geometry optimizations in solution are performed only after an optimized gas phase structure is computed.[21] However, if you only want to obtain an optimized *structure* in solution and do not care about the accuracy of its computed solvation energy, you can skip the gas phase geometry optimization by setting the Gas phase optimization option to no.[22]

# 3.6. PROPERTIES

Various molecular properties can be calculated for a particular wavefunction. These calculations are normally performed using the converged SCF wavefunction. By default, none of the properties listed below are computed, but you can choose to compute them by changing the settings in the Properties window, which can be called up from the main window by clicking on the Properties button.

21.Keyword **nogas** = 0 in **gen** section of input file.
22.Keyword **nogas** = 2 in **gen** section of input file.

## *Electrostatic Potential Fitting*

Jaguar can fit the molecular electrostatic potential (ESP) to a set of point charges [47, 48]. These monopoles can be located either at the atomic centers[23] or at the atomic centers and the bond midpoints,[24] depending on the Fit ESP to selection.

For electrostatic potential fitting of an LMP2 wavefunction, you should also compute a dipole moment for most accurate results, since the charge fitting will then include a coupled perturbed Hartree-Fock (CPHF) term as well. You may also wish to constrain the charge fitting to reproduce the dipole moment, as described below. Since the CPHF term is computationally expensive, it is not included in LMP2 charge fitting by default.

The fit can optionally be constrained to reproduce the dipole moment (and other higher moments, if specified) exactly, by choosing charge + dipole moment[25] (or the appropriate higher-moment choice[26]) from the ESP constraints option menu. (For LMP2 wavefunctions, only dipole moments are available.) Keep in mind that the more constraints you apply to ESP fitting, the less accurately the charge fitting will describe the Coulomb field around the molecule. The dipole moment is generally very close to the quantum mechanical dipole moment as calculated from the wavefunction, and constraining the charge fitting to reproduce it is generally not a problem, but you may obtain poor results if you constrain the fitting to reproduce higher multipole moments. However, this option is useful for some cases, such as for molecules with no net molecular charge or dipole moment.

If both electrostatic potential fitting and multipole moment calculations are performed, the moments will also be computed from the fitted charges for purposes of comparison.

The electrostatic potential is itself computed on a grid. By default, this grid has the same form as the other pseudospectral grids: it is formed by merging sets of spherical shells, whose grid points are centered on each nucleus.[27] An alternative is to use a regular lattice of grid points [48], by choosing rectangular from the ESP grid type option menu.[28] You can then set the spacing in bohr between points

23. Keyword **icfit** = 1 in **gen** section of input file.
24. Keyword **icfit** = 2 in **gen** section of input file.
25. Keyword **incdip** = 1 in **gen** section of input file.
26. Keyword **incdip** = 2 or 3 in **gen** section of input file.
27. Keyword **gcharge** = –1 in **gen** section of input file.
28. Keyword **gcharge** = –2 in **gen** section of input file.

in this lattice in the Rect. grid spacing box.[29] For either grid type, points within the molecular van der Waals surface are discarded. The van der Waals surface used for this purpose is constructed using DREIDING [49] van der Waals radii for hydrogen and for carbon through argon, and universal force field [46] van der Waals radii for all other elements. These values are listed explicitly in Table F near the back of the User's Guide.

You can also print out the values of the ESP at gridpoints whose locations you specify. See the *Properties Keywords* subsection of Section 8.6.

## *Multipole Moments*

Jaguar can compute multipole moments[30] through hexadecapole for HF, GVB, or DFT wavefunctions, and can compute dipole moments for LMP2 wavefunctions. Moments are computed with respect to the center of mass of the molecule. Please note that LMP2 dipole moments can be computationally expensive, since computing them accurately requires coupled perturbed Hartree-Fock calculations.

If you select one of the higher order moments, all moments of lower order will also be calculated. If atomic charges are computed either by fitting of the electrostatic potential [47, 48], as described above, or by Mulliken population analysis [50], as described below, the multipole moments will also be calculated from these point charges for purposes of comparison.

## *Polarizability and Hyperpolarizability*

You can calculate polarizabilities and hyperpolarizabilities by making the appropriate setting in the Polarizability/Hyperpolarizability options menu.

The coupled-perturbed HF option[31] calculates both polarizability and hyperpolarizability using coupled perturbed Hartree-Fock (CPHF) techniques. In general, this option is superior to the finite field option,[32] but the CPHF option can only be used with HF wavefunctions and with basis sets that do not use effective core potentials. (In Jaguar, the basis sets with effective core potentials are CSDZ and those with names beginning with "LA". See Section 3.8 for more information on basis sets.)

29.Keyword **wispc** in **gen** section of input file.
30.Keyword **ldips** = 2, 3, 4, or 5 in **gen** section of input file.
31.Keyword **ipolar** = –1 in **gen** section of input file.
32.Keyword **ipolar** = 5 in **gen** section of input file.

The finite field option [17] uses a five-point finite difference method, which employs the results from thirteen SCF calculations: one with no field; one with an electric field of E (where E is 0.024 A.U. by default) in the x direction; one with a field of –E in the x direction; four others with fields of +E and –E, respectively, in the y and z directions, respectively; and six others using fields of +aE and –aE, respectively, in the x, y, and z directions, respectively, where a is a constant determined automatically.

Both hyperpolarizability methods are run without using molecular symmetry. Also, for any polarizability calculation, the energy convergence criterion, which is set in the Methods window, is set by default to $1.0 \times 10^{-6}$.

If you want to change the electric field used for the finite field calculation or to use other finite field methods to calculate the polarizability and hyperpolarizability, see *Properties Keywords* in section 8.6 for information on editing the input file appropriately.

## *Electron Density*

The electron density for the final wavefunction can be evaluated on a set of grid points. The Cartesian coordinates of these grid points and the electron density in A.U., respectively, for each grid point will appear in the file *jobname*.chdens, where *jobname*.in is the input file for the Jaguar job.

If you select rectangular[33] for the electron density calculation, the grid used will be a rectangular one whose spacing in Angstroms is determined by the density grid spacing[34] set immediately below. For spherical,[35] the default choice, the electron density will be evaluated on the ultrafine grid used by the pseudospectral method. We recommend using the spherical grid for quantitative results, although the cubic grid is sometimes useful for display purposes. (If you wish to use a different grid for electron density calculations, see section 8.6 under *Grid and Dealiasing Function Keywords* for information about the grid keyword **geldens** in the **gen** section of the input file.)

## *Mulliken Population Analysis*

Mulliken populations [50] can be computed for each atom, giving a representation of the molecule as a set of nuclear-centered point charges.[36] For open shell cases, Mulliken spin populations will also

33.Keywords **ldens** = 1 and **geldens** = –3 in **gen** section of input file.
34.Keyword **denspc** in **gen** section of input file.
35.Keywords **ldens** = 1 and **geldens** = 4 in **gen** section of input file.

be computed when Mulliken populations are requested. If you choose to calculate both Mulliken populations and multipole moments, the multipole moments will be computed from the atomic Mulliken populations as well as directly from the wavefunction.

Alternatively, a Mulliken population can be computed either for each of the basis functions as well as for each atom[37], or for each bond between neighboring atoms, as well as by atom and basis function.[38]

### *Natural Bond Orbital (NBO) Analysis*

To request a default Natural Bond Orbital (NBO) analysis [51] at the end of the Jaguar job, turn on NBO analysis[39] at the bottom of the Properties window. The output from the NBO analysis will be included in the Jaguar output file.

Other options for NBO calculations can also be specified in the **nbo** section or in the **core**, **choose**, and **nrtstr** sections of the Jaguar input file. Please note that it is not possible to run NEDA (Natural Energy Decomposition Analysis) calculations from Jaguar, however.

See section 8.19 and the NBO documentation for more details on NBO input and output.

## 3.7. FREQUENCIES & RELATED PROPERTIES

Using the Frequencies window, you can request calculations of frequencies, infrared (IR) intensities, and thermochemical properties (heat capacity, entropy, enthalpy, and Gibbs free energy). Vibrational frequencies and thermochemical properties can be computed for HF, DFT, LMP2, or GVB wavefunctions (except that numerical frequencies cannot be computed for unrestricted HF or DFT wavefunctions). IR intensities are computed by default for frequency jobs for which either a) analytic HF frequencies are computed and the basis set does not have any effective core potentials (see Section 3.8 for details), or b) HF, GVB, or DFT frequencies are computed numerically.

36.Keyword **mulken** = 1 in **gen** section of input file.
37.Keyword **mulken** = 2 in **gen** section of input file.
38.Keyword **mulken** = 3 in **gen** section of input file.
39.Empty **nbo** section in input file ("&nbo &").

## *Frequencies*

You can calculate vibrational frequencies[40] by turning the switch in the Frequencies window to on (compute Hessian). Vibrational frequency calculations are available for HF, GVB, LMP2, and DFT wavefunctions in gas phase or in solution, but are not available for GVB-RCI or GVB-LMP2 calculations.

For gas phase HF and DFT jobs with basis sets that allow pseudospectral calculations and do not include f functions, Jaguar will compute analytic frequencies. (See Section 3.8 for more information on basis sets.) Otherwise, Jaguar will use energies obtained at various perturbed geometries to calculate the numerical derivatives (with respect to coordinates) of the analytically computed forces. (For unrestricted HF or DFT wavefunctions, however, numerical frequencies are not yet available.)

Generally, analytic frequency calculations are much faster than numerical frequency calculations; however, when frequencies are calculated analytically, use of molecular symmetry is turned off for the job. Therefore, if you want to compute analytic frequencies for large, highly symmetric molecules, you should first run any other computationally intensive portions of the job (geometry optimization, for instance), then use the new input (restart) file generated during the job as input for an analytic frequency job. (See section 6.4 for information on generating restart files and restarting jobs.) If, instead, you want to calculate frequencies numerically, you can do so by making the keyword setting **nmder**=2 in the **gen** section of the input file, as described in section 8.6.

To compute frequencies and any requested frequency-related properties from the Hessian available at the end of a job (either an initial Hessian, if it was never updated, or the updated Hessian), choose use available Hessian[41] from the Vibrational Frequencies option menu.

## *Atomic Masses*

For frequency calculations, by default, the atomic mass used for each element is that of its most abundant isotope.[42] However, you can choose to use an average of the isotopic masses, weighted by the abundance of the isotopes, by selecting average isotopic masses[43] from the Atomic masses option menu.

40.Keyword **ifreq** = 1 in **gen** section of input file.
41.Keyword **ifreq** = –1 in **gen** section of input file.
42.Keyword **massav** = 0 in **gen** section of input file.
43.Keyword **massav** = 1 in **gen** section of input file.

## *Scaling of Frequencies*

Because the errors in quantum mechanical calculations of frequencies are often fairly predictable, it is sometimes desirable to scale frequencies by one or more factors. Scaling methods can also improve calculations of thermochemical properties, which use the scaled frequencies. In Jaguar, two different options are available for frequency scaling: Pulay et al.'s Modified Scaled Quantum Mechanical Force Fields (SQM) method [52] for B3LYP calculations using the 6-31G* basis set, and standard frequency scaling, in which all frequencies are simply multiplied by a single parameter.

The SQM method alters the frequencies by scaling the Hessian elements themselves (in internal coordinates), using eleven different scale factors, which depend on the type of stretch, bend, or torsion. This method was parametrized using B3LYP calculations for 30 molecules containing C, H, N, O, and Cl, using the 6-31G* basis set. Jaguar only permits the SQM scaling method to be used for B3LYP/6-31G* frequency jobs. You can turn on SQM scaling for these jobs from the Frequencies window by setting the option menu labeled Pulay SQM scaling to B3LYP 6-31G* factors.[44] The method is off[45] by default.

Alternatively, for any type of frequency job, you can multiply all frequencies by the same scale factor by changing the value in the text box marked Frequency scaling.[46] Table 3.7.1 lists recommended scale factors for various methods and basis sets. The factors in the table are from ref. [53] and are optimized for the best agreement with experiment for the frequencies themselves. Ref. [53] also includes scale factors suitable for use when low-frequency vibrations are of particular interest; for zero-point vibrational energies; and for prediction of enthalpy and entropy. Other scale factors may be available in the literature.

## *Infrared Intensities*

Turn on the IR Intensities[47] setting to calculate infrared intensities for each frequency in km/mol. For HF jobs where frequencies are calculated analytically, the IR intensities are obtained from coupled perturbed Hartree-Fock (CPHF) calculations of the derivative of the dipole moment with respect to changes in the nuclear coordinates, and molecular symmetry is not used for the job. For calculations for

44.Keyword **isqm** = 1 in **gen** section of input file.
45.Keyword **isqm** = 0 in **gen** section of input file.
46.Keyword **scalfr** in **gen** section of input file.
47.Keyword **irder** = 1 in **gen** section of input file.

***Table 3.7.1.*** *Recommended frequency scale factors for various combinations of SCF method and basis set.*

| SCF method | Basis set | Scale factor |
|---|---|---|
| HF | 3-21G | 0.9085 |
| HF | 6-31G* | 0.8953 |
| HF | 6-31+G* | 0.8970 |
| HF | 6-31G** | 0.8992 |
| HF | 6-311G** | 0.9051 |
| MP2 | 6-31G* | 0.9434 |
| MP2 | 6-31G** | 0.9370 |
| MP2 | 6-311G** | 0.9496 |
| BLYP | 6-31G* | 0.9945 |
| BP86 | 6-31G* | 0.9914 |
| B3LYP | 6-31G* | 0.9614 |
| B3P86 | 6-31G* | 0.9558 |
| B3PW91 | 6-31G* | 0.9573 |

which frequencies are computed numerically, the numerical derivative of the dipole can be obtained for IR intensity calculations.

## *Thermochemical Properties*

Thermochemistry calculations of a system's constant pressure heat capacity ($C_p$), entropy (S), enthalpy (H), and Gibbs free energy (G) at standard temperature and pressure are performed by default whenever vibrational frequencies are calculated. Rotational symmetry numbers, which identify the number of orientations of a molecule which can be obtained from each other by rotation, and zero point energies are also computed. You can only calculate these properties if you are also computing vibrational frequencies.

By using the thermochemistry settings, you can control the temperatures and pressure used for calculations of these quantities. The pressure[48] (in atm) used for thermochemical calculations is 1.0 by default, and the initial temperature[49] (in K) is 298.15 by default. Either of these settings can be changed. If you want to compute thermochemical properties at more than one temperature, specify the

48.Keyword **press** in **gen** section of input file.
49.Keyword **tmpini** in **gen** section of input file.

differences between temperatures using the Temp. Increment[50] setting and the number of temperatures at which thermochemical properties should be computed with the Number of steps[51] setting, which is 1 by default.

By default, thermochemical output is in units of kcal/mol (for H and G) and cal/mol K (for $C_p$ and S). To report the output in units of kJ/mol and J/mol K instead, select J/mol from the Output units menu.

# 3.8. BASIS SET

From the Basis Set window, you can choose a basis set[52] from the option menu shown under the label name, and select polarization and diffuse functions from the option menus marked */** and +/++, respectively. If an option is dimmed, it is incompatible with the rest of your input (for instance, the basis set could be missing basis functions for some atom or atoms in your molecule). If you select an italicized basis set name, the calculation will run all-analytically, without using the pseudospectral method. All of these settings are explained further below.

If you do not choose a basis set for a calculation, Jaguar will use the 6-31G** basis set if 6-31G** basis functions are available for all atoms in the input, and will otherwise use the LACVP** basis set by default. These basis sets are described in more detail below.

You can perform counterpoise calculations with Jaguar, adding counterpoise atoms that have the usual basis functions for their elements, but include no nuclei or electrons. However, counterpoise atoms should be input through the Edit window instead of the Basis Set window. Please see Section 2.2 if you wish to use counterpoise atoms.

For any basis set with the ** option, choosing ** places polarization functions on all atoms, unless the basis set uses effective core potentials, in which case polarization functions are placed only on atoms not described with effective core potentials. For STO-3G and 3-21G basis sets, choosing * places polarization functions on all atoms found in the third row (Na-Ar) or higher rows (for STO-3G) of the periodic table. For effective core potential basis sets (CSDZ and those whose names begin with "LA"), choosing * places polarization functions on all atoms not described by effective core potentials except H

50.Keyword **tmpstp** in **gen** section of input file.
51.Keyword **ntemp** in **gen** section of input file.
52.Keyword **basis** in **gen** section of input file.

and He. For all other basis sets, * places polarization functions on all atoms except H and He.

The ++ option places diffuse functions on all atoms, while the + option places diffuse functions on all atoms except H and He. You may find diffuse functions especially useful for calculations on van der Waals complexes or molecules that include atoms with large negative charges.

Table 3.8.1 lists the available basis sets in Jaguar which do not use effective core potentials. The table indicates the atoms these basis sets can describe and shows which sets include the options of polarization and diffuse functions. (To see some of this information in the form of a periodic table, see the basis set tables near the back of the User's Guide.) The cc-pVDZ and cc-pVTZ basis sets include polarization functions by definition. Note that in versions 2.3 and earlier of Jaguar, the cc-pVTZ basis set did not include f functions, and therefore corresponds to the current cc-pVTZ(-f).

Table 3.8.1 also indicates the method used for the calculation: the fast pseudospectral method, or the slower analytic method, in which four-center, two-electron integrals are computed explicitly, as in conventional ab initio programs. The analytic method is used only when optimized pseudospectral grids and dealiasing sets for one or more atoms in the molecule are not yet available. For molecules whose atomic elements are all in the range H-Ar in the periodic table, we recommend using the 6-31G** basis set (the default choice), which is one of the basis sets that permits pseudospectral calculations.

The column headed "# of d fns." indicates whether d shells include the five functions $d_{xy}$, $d_{xz}$, $d_{yz}$, $d_{x^2-y^2}$, and $d_{2z^2-x^2-y^2}$, all with the same angular momentum (l = 2), or whether d shells include the six second-order Cartesian d functions $d_{x^2}$, $d_{y^2}$, $d_{z^2}$, $d_{xy}$, $d_{xz}$, and $d_{yz}$. This choice also affects the dimension of the Fock matrix for diagonalization. If you want to override this selection, you can do so by setting the keyword **numd** in the **gen** section of the input file, as described in the *Basis Set Keywords* subsection of section 8.6. The orbital coefficients are always printed out in terms of the six Cartesian functions. The full references describing the basis sets can be found in the References list at the back of the User's Guide.

The other available basis sets, which are listed in Table 3.8.2, include effective core potentials (ECPs). The names of eight of these basis sets begin with "LA" to indicate they were developed at Los Alamos National Laboratory. If the next character in the name is a "V," the basis set is valence-only, containing only the highest s and p shells for

***Table 3.8.1.*** *Basis sets contained in Jaguar that do not include effective core potentials.*

| Basis Set | Atoms Included | Options | Method | # of d fns. | Refs. |
|---|---|---|---|---|---|
| STO-3G | H-Xe | * (Na-Xe) | analytic | 5 | 54-58 |
| 3-21G | H-Xe | * (Na-Ar), + (Li-Ar), ++ (H-Ar) | H-Ar pseudospectral (except analytic with + or ++), K-Xe analytic | 6 | 59-61 |
| 4-21G | H-Ne | *, ** | analytic | 6 | 62 |
| 6-21G | H-Ar | *, ** | analytic | 6 | 59-61 |
| 4-31G | H-Ne | *, ** | analytic | 6 | 63-68 |
| 6-31G | H-Zn | *, **; +, ++ for H-Ar | H-Ar pseudospectral, K-Zn analytic | 6 | 64-70 |
| 6-311G | H-Ar | *, **, +, ++ | H, Li, C-F, Na, Si-Cl pseudospectral, others analytic | 5 | 71-74 |
| D95V | H, Li-Ne | *, ** | analytic | 6 | 75 |
| D95 | H, Li-Ne, Al-Cl | *, ** | H, Li, C-F, Si-Cl pseudospectral, others analytic | 6 | 75 |
| MSV | H-Ru, Pd-Xe | | analytic | 5 | 76 |
| cc-pVDZ | H-He, B-Ne, Al-Ar | +, ++ | H, C-F, Si-Cl pseudospectral, others analytic; +, ++ analytic | 5 | 77-80 |
| cc-pVDZ(-d) (without d functions) | H-He, B-Ne, Al-Ar | +, ++ | H, C-F, Si-Cl pseudospectral, others analytic; +, ++ analytic | 5 | 77-80 |
| cc-pVTZ | H-He, B-Ne, Al-Ar | +, ++ | H, C-F, Si-Cl pseudospectral, others analytic | 5 | 77-80 |
| cc-pVTZ(-f) (without f functions) | H-He, B-Ne, Al-Ar | +, ++ | H, C-Ne, Si-Ar pseudospectral, others analytic | 5 | 77-80 |

main group atoms and the highest s, p, and d shells for transition metals. For example, 5s and 5p would be included for tellurium, and 6s, 5d, and 6p for tungsten. "LAV1" indicates that the basis set has

been fully contracted to form a minimal basis set, "LAV2" that the last Gaussian has been uncontracted to form a double zeta basis, and "LAV3" that all of the s functions and the last p and d Gaussian have been uncontracted.

Names starting with "LACV" indicate that the basis set also includes the outermost core orbitals (e.g. 5s5p6s5d6p for W). The last letter in each LA basis set name refers to the basis set used for atoms *not* described by ECPs: S indicates the STO-3G basis set, D the D95V basis set, and P the 6-31G set developed by Pople and coworkers. (Note that in addition, for some atoms, the LACVD and LACVP basis sets use the same basis functions as the LAV3D and LAV3P basis sets, respectively.)

Another ECP basis set, CSDZ (for Cundari-Stevens double zeta), extends through the lanthanides. This basis set uses a relativistic effective core potential for the inner core electrons and treats the outer core and valence electrons with a 4s/4p/2d/2f basis set.

Table 3.8.2 describes the ECP basis sets. The atoms described by the effective core potential are listed first, followed by the atoms described by the alternate basis set or sets. (To see some of this information in the form of a periodic table, see Table C and Table D near the back of the User's Guide.) The other table entries provide the same information as that given in the previous table, except that the polarization functions are only applied to atoms obtained from the non-ECP basis sets. All ECP basis sets use five d functions, as described earlier in this section.

Currently, the LACVP, LAV3P, and LACV3P basis sets use the pseudospectral method, while all other ECP basis sets use the analytic method, computing the four-center, two-electron integrals explicitly. We strongly recommend using either the LACVP or the LACV3P basis set for non-lanthanide molecules containing atoms beyond Ar in the periodic table, especially for studies involving charge transfer, $d^0$ metals, or correlated wavefunctions. The LACV3P basis set seems to give substantial improvements over the LACVP basis set for HF, LDA, and B3LYP atomic state splittings. The LACV3P++ basis set, which includes a diffuse d function on any metal atoms, is useful for calculations on anions or low-spin M(0) complexes of the late first row metals.

***Table 3.8.2.*** *Basis sets contained in Jaguar that include effective core potentials.*

| Basis Set | Atoms In ECP | Other Atoms | Options | Refs. |
|---|---|---|---|---|
| LAV1S | Na-La, Hf-Bi | H-Ne (STO-3G) | * (H-Ne) | 81-82 |
| LAV2D | Na-La, Hf-Bi | H, Li-Ne (D95V) | *, ** (H, Li-Ne) | 81-82 |
| LAV2P | Na-La, Hf-Bi | H-Ne (6-31G) | *, ** (H-Ne); +, ++ (H-Ne) | 81-82 |
| LAV3D | Na-La, Hf-Bi | H, Li-Ne (D95V) | *, ** (H, Li-Ne) | 81-82 |
| LAV3P | Na-La, Hf-Bi | H-Ne (6-31G) | *, ** (H-Ne); +, ++ (H-Ne) | 81-82 |
| LACVD | K-Cu, Rb-Ag, Cs-La, Hf-Au | H, Li-Ne (D95V); Na-Ar, Zn-Kr, Cd-Xe, Hg-Bi (LAV3D) | *, ** (H, Li-Ne) | 83 |
| LACVP | K-Cu, Rb-Ag, Cs-La, Hf-Au | H-Ar (6-31G); Zn-Kr, Cd-Xe, Hg-Bi (LAV3P) | *, ** (H-Ar); +, ++ (H-Ar) | 83 |
| LACV3P | K-Cu, Rb-Ag, Cs-La, Hf-Au | H-Ar (6-311G); Zn-Kr, Cd-Xe, Hg-Bi (LAV3P) | *, ** (H-Ar); +, ++ (H-Ar, plus metal diffuse d) | 84 |
| CSDZ | Ce-Lu | H-Ar (6-31G); Zn-Kr, Cd-Xe, Hg-Bi (LAV3P); K-Cu, Rb-Ag, Cs-La, Hf-Au (LACVP) | *, ** (H-Ar); +, ++ (H-Ar) | 85 |

# 3.9. METHODS

The Methods window includes various technical settings which control how the calculation is performed, including the wavefunction type, the source of the initial wavefunction, the convergence method, the maximum number of SCF iterations, and the job's accuracy level. Generally, you should not need to change these options. However, you may want to alter some Methods options if you are having convergence problems.

## *Wavefunction Type (Restricted or Unrestricted)*

To perform an unrestricted HF or DFT calculation, you can select Unrestricted (UHF/UDFT)[53] from the Wavefunction type option

53.Keyword **iuhf** = 1 in **gen** section of input file.

menu. The default method for open-shell systems is restricted open-shell HF or DFT.[54]

## *Choosing an Initial Guess Type*

The default HF initial guess selection is Automatic, meaning that Jaguar selects the initial guess method that is judged to be most likely to lead to convergence. However, you can explicitly select from among several possible initial guess algorithms.

By default, for non-GVB calculations on simple closed-shell systems with no transition metals, Jaguar constructs the initial wavefunction from orbitals that give the best overlap with previously calculated orbitals from atomic calculations.[55] This initial guess method, which can be selected for any calculation, is identified in the interface as AO Overlap. The algorithm used is described in reference [14]. This method compares well with the semi-empirical schemes which other ab initio programs use to obtain initial guesses.

Jaguar also provides a unique initial guess feature to improve SCF convergence (both HF and DFT), particularly for transition-metal-containing systems. As described in reference [19], research at Schrödinger has established that poor convergence of these systems is very often due to problems with the trial wavefunction's orbital shapes and occupations. Schrödinger researchers have therefore developed an algorithm based upon ligand field theory that creates a high-quality initial guess specifically designed for transition-metal-containing systems [19]. The HF initial guess options labeled Ligand Field Theory[56] and LFT + dd repulsion[57] both use this algorithm; see reference [19] for further details on these choices. Both of these initial guess methods can take advantage of user-provided information on charges and spins of "fragments" within the system, as described in section 6.3, although such information is not required.

If you restart a calculation with an input file generated during a previous run, as described in section 6.4, the wavefunction resulting from the earlier run will be read from the **guess** section and used as an initial guess, unless you change the default choice of Read from input.[58] The **guess** section is described in Section 8.11. For versions 2.2 and later of Jaguar, the program can read in an initial guess in one basis set and automatically transform it to the basis set requested for

54.Keyword **iuhf** = 0 in **gen** section of input file.
55.Keyword **iguess** = 10 in **gen** section of input file.
56.Keyword **iguess** = 25 in **gen** section of input file.
57.Keyword **iguess** = 30 in **gen** section of input file.
58.Keyword **iguess** = 1 in **gen** section of input file.

the calculation (unless either basis set uses effective core potentials). If your input file is from version 2.1 or earlier, you should use the same basis set for the restarted calculation or add a basis set label to the **guess** section, as described in Section 8.11.

For GVB calculations, the GVB initial guess options menu lets you choose the method used to generate this guess. By default, the GVB initial guess is automatically constructed from the Hartree-Fock initial guess by piece-wise localization.[59]

Another option is to use a converged HF wavefunction as the basis for the GVB initial guess. For this option, select compute from HF converged wavefunction[60] for the GVB initial guess setting, and make the appropriate setting under HF initial guess for the Hartree-Fock calculation. Because this selection requires two SCF calculations, one for HF and one for GVB, it is considerably more expensive than using the HF initial guess. However, you may wish to try this option if convergence difficulties have been observed.

The third possibility for the GVB initial guess is to read in a GVB wavefunction from the input file's **guess** section and to use that as the initial guess for the calculation.[61] For instance, if you were restarting a job, as described in section 6.4, and wished to use the result from the previous run as an initial guess for the new run, the Read from input option for the GVB initial guess would allow you to do so.

## *Convergence Issues*

We recommend using the default Direct Inversion in the Iterative Subspace (DIIS)[62] or GVB-DIIS[63] SCF convergence schemes [11, 86] whenever possible. The DIIS method generally performs better, but for jobs with SCF convergence problems, GVB-DIIS may give improved convergence. All wavefunctions currently available in Jaguar can be converged with DIIS, including those with multiple open shells and multiple GVB pairs. You may also choose to use a GVB2P5-type convergence scheme, OCBSE[64] [20], although it is generally much slower than DIIS.

You may change the maximum number of SCF iterations allowed.[65] Generally, Hartree-Fock runs for simple organic molecules converge

59. Keyword **ihfgvb** = 2 in **gen** section of input file, or keyword **ihfgvb** = 0 if **iguess** is not 1.
60. Keyword **ihfgvb** = 1 in **gen** section of input file.
61. Keyword **ihfgvb** = 0 and **iguess** = 1 in **gen** section of input file.
62. Keyword **iconv** = 1 in **gen** section of input file.
63. Keyword **iconv** = 4 in **gen** section of input file.
64. Keyword **iconv** = 3 in **gen** section of input file.

in fewer than 10 iterations, while complex calculations using higher-level methods or involving open shells may take a few extra iterations. Molecules which include transition metals may converge more slowly, however.

The default energy convergence criterion for Jaguar,[66] which can also be changed, is set to 5.0 x $10^{-5}$ Hartrees for the total energy on consecutive iterations, except for polarizability calculations, for which it is 1.0 x $10^{-6}$ Hartrees. If the energy difference is less than 1% of the previous energy difference, however, this convergence criterion is overridden for that iteration, and the calculation continues.

When the root mean squared change in density matrix elements for a polarizability, hyperpolarizability, or geometry optimization calculation is less than the RMS density matrix element change criterion,[67] whose default value is 5.0 x $10^{-6}$, the calculation is considered to have converged. For polarizability and hyperpolarizability calculations, if the energy convergence criterion described in the previous paragraph is satisfied first, the calculation will end even if the RMS density matrix element change criterion has not been met, and vice versa.

The SCF level shift[68] setting describes the amount the virtual orbitals' energies are increased before diagonalization, in atomic units. This setting can be used to reduce mixing of the real and virtual orbitals, which can slow convergence, but often helps otherwise intractable cases to converge. Useful SCF level shift values are generally in the range 0.3–1.0.

## *Accuracy Level*

The grids used for various SCF iterations and the accuracy with which parts of the calculation are done greatly affect the timing, and sometimes the accuracy, of the entire calculation. You can adjust the grids and the set of cutoff values determining these factors by using the Accuracy level option menu. The usual default quick setting[69] allows fast calculations to be performed, using several different pseudospectral grid types, and cutoffs which should generally produce well-converged energies. The accurate setting,[70] which corresponds to somewhat stricter cutoffs (and therefore somewhat slower calculations), also uses a variety of pseudospectral grids.

65.Keyword **maxit** in **gen** section of input file.
66.Keyword **econv** in **gen** section of input file.
67.Keyword **dconv** in **gen** section of input file.
68.Keyword **vshift** in **gen** section of input file.
69.Keyword **iacc** = 3 in **gen** section of input file.
70.Keyword **iacc** = 2 in **gen** section of input file.

If you choose the ultrafine setting,[71] the cutoffs will be "tight" (very accurate), and only the ultrafine pseudospectral grid type will be used. The ultrafine setting may be helpful for cases with convergence or accuracy problems, but will increase the computational cost by a factor of two to three.

More information on grids and cutoffs can be found in the descriptions of the grid and cutoff files, in section 9.4 and section 9.5, respectively.

## *Analytic Corrections*

For efficiency reasons, Jaguar uses both numerical and analytical methods. There is a trade-off required: analytic methods are more accurate, but also more computationally expensive. Setting analytic corrections to Fully analytic calc.[72] results in a non-pseudospectral calculation, which will be significantly slower than the usual method. The default is Standard Corrections,[73] in which case the exact number and type of analytically calculated two-electron integrals [87], [88] are automatically determined.

## *Final Localization of the Orbitals*

By default, the final wavefunction is not localized.[74] You may localize the valence orbitals after the wavefunction is computed with either the Boys[75] procedure [38] or the Pipek-Mezey[76] procedure [39]. If you choose Boys as a Final localization method, Jaguar localizes the doubly-occupied orbitals by maximizing the term $\Sigma_{ij}|\langle\varphi_i|r|\varphi_i\rangle - \langle\varphi_j|r|\varphi_j\rangle|^2$. Pipek-Mezey localization is performed by maximizing the sum of the squares of the atomic Mulliken populations for each atom and occupied orbital. See section 5.7 to find out how to print the localized orbitals resulting from either method.

Both of the available localization methods scale as $N^3$ with basis set size. However, the use of molecular symmetry is turned off for the entire job whenever you perform a final localization, so for fastest results you may wish to run a job without localization, then restart the job after turning on localization in the new input file. See section 6.4 if you need information on restart files and restarting jobs.

71. Keyword **iacc** = 1 in **gen** section of input file.
72. Keyword **nops** = 1 in **gen** section of input file.
73. Keywords **noatcor** = 0 and **nops** = 0 in **gen** section of input file.
74. Keyword **locpostv** = 0 in **gen** section of input file.
75. Keyword **locpostv** = 1 in **gen** section of input file.
76. Keyword **locpostv** = 2 in **gen** section of input file.

## *Symmetry*

By default, Jaguar takes advantage of molecular symmetry[77] whenever possible, in order to obtain CPU savings. Both Abelian and non-Abelian point groups are recognized (a particular strength of Jaguar). If you wish, however, you can turn the use of symmetry off.[78] For information on how to make sure the symmetry of your input structure will be treated as you expect, please see Section .

For some calculations, including GVB, LMP2, GVB-LMP2, and GVB-RCI calculations and calculations of IR intensities or hyperpolarizabilities, symmetry is not yet implemented and will be disabled automatically during the job.

77.Keyword **isymm** = 8 in **gen** section of input file.
78.Keyword **isymm** = 0 in **gen** section of input file.

# 4. Optimizations and Scans

For Hartree-Fock, GVB, LMP2, and DFT calculations in gas phase or in solution, Jaguar can use calculated analytic gradients to optimize the molecular geometry to a minimum-energy structure or a transition state, while for GVB-RCI calculations, optimizations can be performed using numerical forces.

Throughout this chapter, footnotes indicate the Jaguar input file keywords and sections that correspond to particular interface settings. If you are working from the interface, you can ignore these footnotes, but you may find them helpful if you decide to use input files to submit jobs without using the interface or if you want to edit keywords directly by using the Edit Job window described in section 2.8 under *Editing Input*.

## 4.1. Geometry Optimization: The Basics

The Optimization window contains the settings for optimization of minimum-energy structures or transition states. If you want to constrain bond lengths or angles during an optimization, you may also wish to use the Edit window, as described in Section 4.2.

The Optimization task option menu is set to none[1] by default, meaning Jaguar will perform a single-point calculation. The energy minimization option[2] requests a search for the molecular geometry with the lowest possible energy. The calculate forces only[3] option in the Optimize geometry option menu computes the derivatives of the

1. Keyword **igeopt** = 0 in **gen** section of input file.
2. Keyword **igeopt** = 1 in **gen** section of input file.

energy for the input structure but does not actually change the geometry. Another option is transition state search.[4] Because the settings for transition state searches are considerably more complex than for geometry optimizations to minimum-energy structures, settings specific to transition state optimizations will be described in a later section of this chapter.

## *Maximum Iterations (Number of Structures Generated)*

An upper limit on the number of steps taken in the geometry optimization sequence—that is, the number of structures generated during the run—can be set in the box marked Maximum iterations.[5] The default is 100. Many cases will meet the convergence criteria after ten or fewer geometries are computed. However, input containing very floppy molecules, transition metal complexes, poor initial geometries, or poor initial Hessians may require many more cycles, and in particularly bad cases, may also require you to stop the calculation and restart it with a change in one or more of the other default Optimization settings described below.

## *Geometry Convergence Issues*

For optimizations to minimum-energy structures or transition states, the convergence criterion for SCF calculations is chosen to assure accurate analytic gradients. For these jobs, a wavefunction is considered converged when the root mean squared change in density matrix elements is less than the RMS density matrix element change criterion,[6] whose default value is $5.0 \times 10^{-6}$. (The SCF calculations during an optimization to a minimum-energy structure or transition state do not use the energy convergence criterion used by other SCF calculations.) The RMS density matrix element criterion may be set in the Methods window, whose button appears in the main interface window.

For initial iterations of an optimization, the SCF calculations are performed at the quick accuracy level described in section 3.9 under *Accuracy Level* (unless the input contains a transition metal, in which case the accuracy level is accurate). However, for the last few geometry iterations, the accuracy level for the SCF calculations is reset to the accurate level, and the SCF calculations thus use tighter cutoffs and denser pseudospectral grids than at the quick level.

3. Keyword **igeopt** = –1 or keyword **nmder** = 1 in **gen** section of input file, depending on whether analytical or numerical forces are requested.
4. Keyword **igeopt** = 2 in **gen** section of input file.
5. Keyword **maxitg** in **gen** section of input file.
6. Keyword **dconv** in **gen** section of input file.

The geometry is considered to have converged when the energy of successive geometries and the elements of the analytic gradient of the energy and the displacement have met the convergence criteria. These criteria are all affected by the Convergence criteria choices, default[7] or loose;[8] the loose criteria are all five times larger than the default criteria. For optimizations in solution, the default criteria are multiplied by a factor of three, and a higher priority is given to the energy convergence criterion. Thus, if the energy change criterion is met before the gradient and displacement criteria have been met, the geometry is considered converged.

See *Geometry Optimization and Transition State Keywords* in section 8.6 if you want more details on the geometry optimization convergence criteria or information on how to edit the input file to set them directly.

## *The Initial Hessian*

To perform an optimization, Jaguar first needs to read or generate an initial Hessian (second derivative matrix or force constant matrix).

You can provide Jaguar with a Hessian in the **hess** section of an input file, as described in section 8.10. For instance, if you restart a geometry optimization from a previous job, as described in section 6.4, Jaguar will automatically use the Hessian provided in your input file.

If your input file does not contain a Hessian, you can use the Initial Hessian option menu in the Optimization window to specify what kind of initial Hessian Jaguar should generate. You can select from among several internal guesses: the Fischer-Almlöf Hessian[9] [42], the Schlegel Hessian[10] [43], or the unit matrix.[11] For most cases, the Schlegel or Fischer-Almlöf options are the best choices. The Schlegel guess is the default.

The final option, quantum-mechanical,[12] is to have Jaguar compute the initial Hessian. This calculation is the most time-consuming of the initial Hessian options. Theoretically, it can be the best option for cases where the other Hessian choices are inadequate, although in practical terms, other steps taken to improve optimizations are likely to be more cost-effective.

7. Keyword **iaccg** = 2 in **gen** section of input file.
8. Keyword **iaccg** = 3 in **gen** section of input file.
9. Keyword **inhess** = –1 in **gen** section of input file.
10. Keyword **inhess** = 0 in **gen** section of input file.
11. Keyword **inhess** = 1 in **gen** section of input file.
12. Keyword **inhess** = 4 in **gen** section of input file.

### *Trust Radius*

In order to avoid changing the geometry too much because of an unusually shaped potential well or an inaccuracy in the Hessian, Jaguar restricts the norm of the changes to the Cartesian or internal coordinates to be less than a certain trust radius, which is defined in atomic units (bohr and/or radians). The trust radius can sometimes vary from one iteration to another.

If the trust radius is marked Fixed,[13] the trust radius will remain the same throughout the optimization (except when Jaguar determines that changing it will lead to better convergence for problem jobs). This setting is the default for optimizations to minimum-energy structures. If the trust radius is not fixed[14] (the default for transition state optimizations), Jaguar will keep geometry changes within the region that is well-described by the Hessian by increasing the trust radius when the Hessian is correctly predicting energy changes and decreasing it when the predictions are inaccurate.

At the beginning of a job, the trust radius starts at the value in the box marked Trust radius. For gas phase optimizations to minimum-energy structures, by default, the trust radius starts at 0.3 atomic units; for solvation calculations or transition state optimizations, the trust radius starts, by default, at 0.1 atomic units.

## 4.2. CONSTRAINING COORDINATES

To constrain certain coordinates to stay frozen (unchanged) or equal to each other during an optimization, you can use the Optimization window settings or the Edit window. The first subsection of this section describes how to freeze an entire class of coordinates during an optimization. The other subsections describe ways to constrain individual coordinates by editing the geometry input.

### *Freezing All Bond Lengths, Bond Angles, or Torsional Angles*

You can freeze all bond lengths,[15] all bond angles,[16] and/or all torsional angles[17] during an optimization. Jaguar will then keep these coordinates unchanged throughout the job.

13. Keyword **itradj** = 0 in **gen** section of input file.
14. Keyword **itradj** = 1 in **gen** section of input file.
15. Keyword **nooptr** = 1 in **gen** section of input file.
16. Keyword **noopta** = 1 in **gen** section of input file.

To freeze all coordinates of a particular type, first open the Optimization window. To make Constraints settings from this window, you must first choose energy minimization or transition state search from the Optimization task option menu. You can then turn any of the constraint settings for Bond Lengths, Bond Angles, or Torsional Angles using the buttons at the bottom of the Optimization window.

## *Freezing Specific Coordinates*

You can constrain (freeze) one or more specific coordinates in your geometry input[18] to remain fixed at its original value during an optimization. From the interface's Edit Geometry or Edit Job window, you can freeze a specific coordinate by adding a '#' sign at the end of its value in your geometry input.

For example, to fix the HOH bond angle of water to be 106.0 degrees, you could input the following Z-matrix:

```
O
H1  O   0.9428
H2  O   0.9428   H1   106.0#
```

If you performed a geometry optimization on this input geometry, the bond angle would remain frozen at 106° throughout the optimization, although the bond lengths would vary.

To freeze any variables to remain unchanged during an optimization, add a '#' sign to the end of the variable setting in the line at the end of the geometry input that defines the variables, as in this example, where the C–H bond is frozen at 1.09 Å:

```
chbond=1.09#  HCHang=109.47
```

Alternatively, you can freeze the values of individual coordinates set with variables without directly constraining the variables. For example, if your input for an optimization of a water molecule looked like this:

```
O   0.000000     0.000000  -0.113502
H1  0.000000     ycoor      zcoor#
H2  0.000000    -ycoor      zcoor#
ycoor=0.753108   zcoor=0.454006
```

the H atoms would only be allowed to move within the xy plane in which they started.

17. Keyword **nooptt** = 1 in **gen** section of input file.
18. The geometry input, including constraints (# signs), is in the **zmat** section of the input file.

If frozen Cartesian coordinates are included in the input for an optimization, Jaguar uses Cartesian coordinates for the optimization rather than generating redundant internal coordinates, and the optimization does not make use of molecular symmetry.

## *Applying Constraints by Using Variables*

If you use the same variable to set several coordinates in your geometry input, Jaguar will change the value for that variable during the optimization, and each coordinate set to that variable originally will then have the same new value as the variable does. Thus, by using variables in your input, you can require coordinate values to remain equal to each other (or equal except for sign) throughout an optimization, even while the coordinates change from their original values.

The effect of using variables depends upon the format of your input structure. If your input structure is in Z-matrix format, you can set several bond length or angle coordinates to the same variable, thus requiring them to have the same values as each other during an optimization even while they change. For input in Cartesian format, you can use variables to keep several atoms within the same plane during an optimization by setting their coordinates along one axis to the same variable.

To use variables to set coordinate values from the interface's Edit Geometry or Edit Job window, first type the variable name (zcoor, for instance) where you would normally type the corresponding numerical value for each relevant coordinate. You can put a + or – sign immediately before any variable, and you may use several variables if you wish. When you have entered the full geometry, add one or more lines setting the variables.

For instance, the Cartesian input

```
O   0.000000     0.000000  -0.113502
H1  0.000000     ycoor      zcoor
H2  0.000000    -ycoor      zcoor
ycoor=0.753108   zcoor=0.454006
```

describes an input structure for water. If you performed a geometry optimization using this input structure, its ycoor and zcoor values might change, but their values for one hydrogen atom would always be the same as those for the other hydrogen atom; therefore, the H atoms in the final geometry would be in the same xy plane as each other.

Whenever Cartesian input with variables is used for an optimization, Jaguar performs the optimization using Cartesian coordinates rather than generating redundant internal coordinates, and the optimization does not make use of molecular symmetry.

# 4.3. TRANSITION STATE OPTIMIZATIONS

To perform transition state searches with Jaguar, you can use either a simple quasi-Newton method that searches for the transition state nearest to the initial geometry, or quadratic synchronous transit (QST) methods (also known as synchronous transit quasi-Newton (STQN) searches). We generally recommend using QST methods any time you can provide both reactant and product geometries.

From the Jaguar interface, the first step in setting up a transition state search is to open the Optimization window and select transition state search[19] from the Optimization task option menu. As soon as you make this selection, the Transition state search window will open automatically. You should use this window to describe the sort of transition state optimization you wish to perform, then return to the Optimization window to set any other optimization features that are not unique to transition state searches.

This section describes various transition state search options in Jaguar. For information on general settings that are useful for all types of geometry optimizations, see Section 4.1.

## *Transition State Search Method*

The first choice listed in the Transition state search window is the Search method, which can be set to standard[20] or QST-guided.[21] The default choice is standard because it does not require more than one input geometry, but if you can provide product and reactant geometries, we recommend selecting QST-guided.

For a QST-guided search, you must input either two geometries, corresponding to the reactant and the product, or three geometries, corresponding to the transition state guess, the reactant, and the product. (The distinction between reactant and product is arbitrary for Jaguar.) The subsection *Specifying Different Structures for the Reaction* below describes how to input these structures. If you provide

19.Keyword **igeopt** = 2 in **gen** section of input file.
20.Keyword **iqst** = 0 in **gen** section of input file.
21.Keyword **iqst** = 1 in **gen** section of input file.

reactant and product geometries, but not a transition state guess, Jaguar will generate a transition state guess by interpolating between these two structures; see the subsection *Initial LST Guess* below for more details.

For each of the first few steps of a QST-guided search, the optimizer is restricted to search along the circular curve connecting the reactant, transition state guess, and product structures. This restriction prevents the optimizer from being led far astray by the inaccuracies of the guess Hessian, and prevents it from exploring transition states that do not correspond to the reaction of interest. During these steps, the optimizer approaches (or finds) the maximum-energy structure along the reactant-to-product curve, and also greatly improves the Hessian.

Once it has obtained the improved Hessian and transition state guess, the optimizer removes the strict requirement that the search must be along the circular curve between the structures. For all subsequent steps in the search, the optimizer follows the Hessian eigenvector that is most similar to the tangent of the circular curve. (If no Hessian eigenvector is sufficiently similar to the tangent to the curve, the optimizer follows the lowest eigenvector.)

If you have a fairly good transition state guess but cannot provide reactant or product structures, you can still use the standard, non-QST method. This optimizer attempts, at each step, to maximize the energy along the lowest-frequency eigenvector of the Hessian and to minimize along all other coordinates. This process is well-defined and straightforward when the Hessian has exactly one negative frequency, indicating that the structure is near a saddle point. The negative-eigenvalue mode, which is sometimes known as the *reaction coordinate,* is referred to as the *transition vector* in this chapter.

## *Specifying Different Structures for the Reaction*

As mentioned above, for a QST-guided search, you must input either two geometries, corresponding to the reactant and the product, or three geometries, corresponding to the transition state guess, the reactant, and the product. In either case, for best results, the reactant and product structures should not be radically different from the transition state. For instance, to find the transition state in a bond-breaking reaction, it would be better to provide a product structure in which the breaking bond was fairly long and weak than a true minimum-energy structure in which the bond had completely dissociated.

If you are inputting three geometries, the transition state guess is the main geometry used, so the reactant and product geometries are

labeled Geometry 2 and Geometry 3. If you input two geometries, the first geometry is the main geometry used, and the second is considered to be Geometry 2. The main geometry for the job is the one used to determine constraints and coordinates. Constraints in Geometry 2 or Geometry 3 are ignored, and the atoms must be listed in these geometries in the same order in which they appear in the main geometry.

To input the main geometry, use the usual procedures: either read it into the interface, or create it within the Edit window. To input or edit Geometry 2 or Geometry 3, you should also use the Read window or the Edit window, with a few minor changes. To read in a geometry as Geometry 2 or Geometry 3, set the Read as option menu in the Read window to Geometry 2 or Geometry 3. To create or edit one of these geometries from the Edit Geometry window, choose Geometry 2 or Geometry 3 from the Structure pull-down menu.

Also, from the Edit Geometry window, note that you can use the same Z-matrix for Geometry 2 or Geometry 3 as you are using for the main geometry, merely defining variables to have different settings. To do so, set the Structure menu to Geometry 2 or Geometry 3, then select Use initial geometry Z-matrix from the Z-matrix pull-down menu.

You can display additional geometry or geometries for a QST-guided search in the Display window. Simply select Geometry 2 or Geometry 3 from the Structure pull-down menu.

## *Initial LST Guess*

As mentioned above in the subsection *Transition State Search Method*, if you provide reactant and product geometries for a QST-guided search, but do not provide a transition state guess, Jaguar will generate a transition state guess by interpolating between these two structures.

By default, this linear synchronous transit (LST) transition state guess will be midway between the reactant and product geometries. This choice is indicated in the Jaguar interface by the default value of 0.5 for the Initial LST guess[22] setting. If you wish to pick a transition state guess closer to the reactant geometry, you can change this setting to a number between 0 and 0.5; to pick a guess closer to the product geometry, set the Initial LST guess value to a number between 0.5 and 1.0.

22.Keyword **qstinit** in **gen** section of input file.

## *Searching Along a Particular Path or Eigenvector*

If you are using the standard (non-QST-guided) optimization transition state optimization method, you can use the Search along option menu to specify a path for the optimizer to follow or an eigenvector for it to minimize each iteration: the lowest Hessian eigenvector[23] (the default), the lowest non-torsional mode,[24] the lowest bond-stretch mode,[25] the reactant-product path,[26] or a user selected eigenvector.[27]

Under certain circumstances, you may want to direct your transition state search using these options, rather than having the optimizer simply minimize along the lowest Hessian eigenvector found for each iteration. The lowest non-torsional mode and lowest bond-stretch mode options can be useful for steering the optimizer to a particular type of transition state—for instance, for a study of a bond-breaking reaction, you can avoid converging to a torsional transition state by choosing lowest bond-stretch mode. The reactant->product path option causes the optimizer to follow the Hessian eigenvector that is most similar to the direction of the linear path between the reactant and product structures (if you have provided these two structures, but no transition state guess, in your input) or the tangent of the circular curve between the reactant, transition state guess, and product structures (if you have provided all three of these structures in your input).

If you know the number of the eigenvector along which you would like to minimize (a particular bond stretch, for instance), you can make the optimizer follow that eigenvector by setting Search along to user selected eigenvector and specifying the eigenvector number in the text box marked Selected eigenvector.[28] You can identify the eigenvector number by running one geometry optimization iteration (see *Maximum Iterations (Number of Structures Generated)* in Section 4.1 for help) and examining the output summary of the Hessian eigenvectors, which indicates the dominant internal coordinates and their coefficients for each eigenvector.

23. Keyword **itrvec** = 0 in **gen** section of input file.
24. Keyword **itrvec** = –1 in **gen** section of input file.
25. Keyword **itrvec** = –2 in **gen** section of input file.
26. Keyword **itrvec** = –5 in **gen** section of input file.
27. Keyword **itrvec** > 0 in **gen** section of input file.
28. Keyword **itrvec** > 0 in **gen** section of input file, where **itrvec** is the relevant eigenvector number for the selected eigenvector.

## *Eigenvector Following*

The setting for Eigenvector following determines whether a minimization will follow a new eigenvector each iteration[29] (the default behavior, with eigenvector following off) or whether the transition state optimizer will follow the eigenvector that most closely correlates with the one chosen the previous iteration.[30]

## *Refinement of the Initial Hessian*

The quality of the Hessian in the initial steps of a transition state optimization can have a marked effect on the speed of the job, since the Hessian controls the direction Jaguar travels on a potential energy surface in its search for an appropriate saddle point. The QST-guided transition search method effectively refines the Hessian automatically in the first steps along the circular curve connecting the reactant, transition state guess, and product.

With the standard, non-QST-guided optimization method, if a coordinate with a negative force constant (Hessian eigenvalue) exists, it is critical for this transition vector to be properly identified as efficiently as possible, since it leads to the transition state. Consequently, for transition state searches with the standard optimizer, when the initial Hessian chosen is a guess Hessian (one not calculated numerically or read from a restart file), it can be helpful to refine the Hessian during the calculation before using it to compute any new geometries.

Hessian refinement is especially likely to improve transition state optimizations that employ eigenvector following (described under *Eigenvector Following* in this section), because any eigenvector selected for following should be accurate enough to be a reasonable representation of the final transition vector.

To refine an initial Hessian, first choose low-frequency modes from the Hessian refinement option menu in the Transition state search window, which should open when you select transition state search in the Optimization window's first option menu.

Next, you must specify the number of low-frequency Hessian eigenvectors to be used in the refinement, or you must specify one or more input coordinates for refinement. (You can also use both low-frequency modes *and* particular coordinates for a refinement.) The subsection *Specifying Coordinates for Hessian Refinement* explains how to select input coordinates for refinement. If you want to specify

29.Keyword **ifollow** = 0 in **gen** section of input file.
30.Keyword **ifollow** = 1 in **gen** section of input file.

a certain number of low-frequency eigenvectors, edit the number in the box marked # of low-freq modes.[31] (By default, no eigenvectors are used—that is, no refinement is performed unless the input specifies particular coordinates for refinement.) Hessians can be refined using any number of the lowest-frequency Hessian eigenvectors. Refinements involve SCF and gradient calculations for displacements along these modes, which allow more accurate information about the most important modes to be included in the Hessian.

## *Specifying Coordinates for Hessian Refinement*

If you are optimizing a molecular structure to obtain a minimum-energy structure or a transition state, you may wish to refine the Hessian used for the job. (Whether or not you refine particular coordinates, you can specify a certain number of the lowest eigenvectors of the Hessian for refinement, as described in the subsection *Refinement of the Initial Hessian*—the Hessian can be refined in both ways in the same job.)

If you put an asterisk ('*') after a coordinate value, Jaguar will compute the gradient of the energy both at the original geometry and at a geometry for which the asterisk-marked coordinate has been changed slightly, and will use the results to refine the initial Hessian to be used for the optimization. To request refinement of a coordinate whose value is set using a variable, add an asterisk to the end of the variable setting in the line at the end of the geometry input that defines the variables. For instance, if you entered either of the following two input geometries in the Edit window:

```
O1
H2   O1   1.1*
H3   O1   1.1*   H2   108.0*
```

or

```
O1
H2   O1   ohbond
H3   O1   ohbond   H2   108.0*
ohbond = 1.1*
```

they would have the same effect: a job from either input that included Hessian refinement would use both O–H bonds and the H–O–H angle in the refinement.

Molecular symmetry or the use of variables, either of which may constrain several coordinate values to be equal to each other, can

31. Keyword **irefhup** = 3 in **gen** section of input file.

reduce the number of coordinates actually used for refinement. For example, for the second water input example shown above, only two coordinates will actually be refined (the O–H bond distance, which is the same for both bonds, and the H–O–H angle); the same would be true for the first example if molecular symmetry is used for the job.

# 4.4. GEOMETRY SCANS

Geometry scans are a series of jobs run with input files that vary only in the value of one or more particular variables used to define an internal or Cartesian coordinate in the input structure. For instance, if you want to perform a "relaxed scan," finding minimum-energy geometries while holding a particular coordinate fixed to various values, you can simply set up a geometry optimization input file with a description of the values that coordinate should take on. As another example, you could calculate the solvation energy of the staggered and eclipsed conformations of ethane by submitting an input file that specified that the torsional angle about the C–C bond take on the two different values for the two structures.

If you are going to vary a particular coordinate for a scan, you can dictate the values it takes on by listing them explicitly or by specifying its initial value and any other two of the following factors in the order listed here, where *integer* means an appropriate integer and *number* means an appropriate real number:

- the initial value of the coordinate (its value for the first job), specified just by number or by "from *number*"
- the final value of the coordinate (its value for the final job), specified by "to *number*"
- the step size (the amount the coordinate changes from one job to the next), specified by "by *number*"
- the number of steps (the number values you want the coordinate to have), specified by "in *integer*"

Note that the initial and final values are always among the values set. Therefore, varying a coordinate from 0 to 120 by a step size of 30, for example, would actually take 5 steps: 0, 30, 60, 90, and 120.

To scan over a particular coordinate, just set that coordinate with a variable in the geometry input (as described in section 2.2), then set the variable using three of the four formats indicated in the list above. For instance, to vary the angle HCCH over the values {0, 30, 60, 90, 120, 150, 180}, you could set it with any one of the following lines:

```
HCCH = from 0 to 180 by 30
HCCH = 0 to 180 in 7
HCCH = from 0 by 30 in 7
```

You can also set a coordinate to a set of specific values using the word "at". With the "at" format, the values of the scanned coordinate do not have to be evenly spaced. For example, this line would vary the angle HCCH over the values {0, 30, 60, 70, 80, 90, 120, 150}:

```
HCCH = at 0 30 60 70 80 90 120 150
```

You can define up to five scan coordinates at once. The first scan coordinate will be the innermost on the loop—that is, the scanner will run through all values of the first scan coordinate before updating the others, and so on, finally looping last over the last scan coordinate.

An additional output file with the name <jobname>.steps.in is written to the working directory whenever a scan is performed. This file contains the geometry specifications for each geometry in the scan, along with the calculated energies, keywords, and forces.

# 5. Output

The output from a Jaguar run always includes a Jaguar output file, which contains the primary output, and a log file, which is mainly useful as a job summary as the job is being run. If you request other output options from the Files window, whose button is found under the Output heading, various other files can also be generated as output.

This chapter begins with a description of the Jaguar output file for a standard Hartree-Fock calculation, continuing with a discussion of the changes in the output for various other calculation options and the output options which can be set from the Standard, Files, Per Iteration, and Orbitals windows found under the Output heading. The final section explains the log file, which is the file displayed when the Check Job button in the interface's file viewer window is clicked as a job runs.

Throughout this chapter, footnotes indicate the Jaguar input file keywords and sections that correspond to particular interface settings. If you are working from the interface, you can ignore these footnotes, but you may find them helpful if you decide to use input files to submit jobs without using the interface.

## 5.1. SUMMARIZING JAGUAR RESULTS

If you can run perl scripts on your system, you can obtain summaries of Jaguar results in simple table form by using the command "jaguar results". Jaguar will then search output files for the information you request through "jaguar results" options.

The tables produced by "jaguar results" can describe results from one job or several jobs. The results can be restricted to final results from each job listed (the default), or can include intermediate results (SCF energies for each geometry in an optimization, for instance). By default, each line lists information that pertains to the entire input structure, but you can also request some kinds of information for each individual atom in the structure. Each of these types of "jaguar results" tables are described below.

You can use "jaguar results" tables to summarize key output in an easy-to-read format or to produce files in a particular format for use in other scripts or programs you write. For either of these purposes, we recommend creating aliases for the "jaguar results" commands you find most useful, and, if you want, writing the "jaguar results" output to a file for later viewing or use. If you need help accomplishing either of these tasks, please contact Schrödinger or your system administrator.

## *Reporting Final Results from One or More Jobs*

By default, each row of the "jaguar results" table (except the title row) corresponds to the final results from a Jaguar output file that was listed in the "jaguar results" command. For instance, if you entered the command

```
jaguar results -energy RuCp2.out piperidine.out
```

from a directory containing the output files RuCp2.out and piperidine.out, you would get a very simple table like this:

```
  -480.726524
  -250.470399
```

where the first line lists the final energy from the job RuCp2 and the second lists the final energy from the job piperidine.

If your "jaguar results" command includes the flag "-title", the table will have column headings indicating the type of information listed. For instance, if you modified the command above to include the "-title" flag:

```
jaguar results -title -energy RuCp2.out piperidine.out
```

your "jaguar results" table would look like this:

```
    Energy
   [hartree]
============
 -480.726524
 -250.470399
```

(If you want to see ahead of time what the column headings of your table would look like without any results listed, use the "-titleonly" flag.)

The "jaguar results" tables can list both information describing the job run (for instance, its name, the basis set and SCF method used, or the stoichiometry of the molecule) and information about the results of the job (for example, the final energy or dipole moment). Each of these types of information appears in a column in the table. If you use the "-title" flag, column labels will indicate the type of information in each column. The columns appear in the table in the same order they are listed in the "jaguar results" command.

Table 5.1.1 and Table 5.1.2 show some options you can use to get "jaguar results" tables summarizing final results of a job. Table 5.1.1 shows options you can use to get a description of the job run (as determined by your input file); Table 5.1.2 lists options that help you obtain actual final calculated properties and results.

***Table 5.1.1.*** *"jaguar results" options that show information about the job run.*

| **jaguar results flag** | **corresponding jaguar results column output** |
|---|---|
| jobname | job name |
| longjobname | job name, with wider output |
| method | SCF/post-SCF method |
| basis | basis set |
| stoich | stoichiometry of input structure |
| weight | molecular weight of input structure |
| charge | molecular charge |
| multip | spin multiplicity |
| nbasis | number of basis functions |
| natom | number of atoms |
| nelectron | number of electrons |
| symmetry | molecular symmetry |
| nsymm | symmetry number |

***Table 5.1.2.*** *"jaguar results" options that show information about final (or intermediate) results from the job.*

| jaguar results flag | corresponding jaguar results column output |
|---|---|
| energy | final molecular energy |
| enuc | nuclear repulsion energy |
| egas | gas-phase energy |
| esoln | solution-phase energy |
| esolv | solvation energy |
| zpe | zero-point energy |
| zvar *zvarname* | value for Z-variable *zvarname* |
| dipole | total dipole moment |
| time | total cpu time for job |

The order of the list of "jaguar results" options determines the order of the columns of information in the table (from left to right). For instance, the command

```
jaguar results -title -jobname -method -energy h2o.out h2o_b3lyp.out
```

(where h2o.out and h2o_b3lyp.out are output files from jobs at the Hartree-Fock and density functional theory (with the B3LYP functional) levels, respectively) gives the "jaguar results" table

```
     Jobname          Method        Energy
                                   [hartree]
 ================  ===========  =============
 h2o                    HF          -76.023641
 h2o_b3lyp             B3LYP        -76.418721
```

with the job name, method, and energy listed from left to right in the same order they were in the "jaguar results" command.

## *Reporting Intermediate Results*

By default, only the final results are reported for each job; therefore, for instance, a table of results from three jobs would have three rows of information. However, you can also request that information from each geometry, SCF, or gradient calculation be reported in a different row of the "jaguar results" table. For instance, the command

```
jaguar results -title -all -iterg -echange -gmax -grms -dmax -drms dftg.out
```

here produces a table showing the convergence of a BLYP geometry optimization of water:

```
Geopt    Energy       Gradient     Gradient     Displace.    Displace.
iter    [change]        [max]        [rms]        [max]        [rms]
====  ===========  ===========  ===========  ===========  ===========
  1                 3.22E-02 .   2.65E-02 .   5.53E-02 .   4.88E-02 .
  2   -2.04E-03 .   3.85E-03 .   3.18E-03 .   2.79E-02 .   1.70E-02 .
  3   -7.04E-05 .   4.19E-04 *   3.82E-04 .   1.45E-03 *   1.01E-03 *
  4   -1.04E-06 #   3.05E-05 #   2.52E-05 #   6.13E-05 #   5.13E-05 #
```

Table 5.1.3 lists the "jaguar results" options that let you specify when

***Table 5.1.3.*** *"jaguar results" options that let you request intermediate results from a job as well as final results.*

| jaguar results flag | corresponding jaguar results row output |
|---|---|
| all | report results every geometry iteration |
| allscf | report results for each scf |
| allder1b | report results for each der1b |

to report intermediate (and final) results from jobs. The "-all" option, which lets you track the progress of a geometry or transition state optimization, is likely to be the most useful of the options shown in Table 5.1.3. The "-allscf" option can be used for intermediate results in complex non-optimizations, such as solvation jobs.

You can use the options from Table 5.1.3 in combination with any of the "jaguar results" options described earlier, such as job name or energy results (see Table 5.1.1 and Table 5.1.2). You can also report various types of output that are primarily useful for reporting for the intermediate steps of the job, as shown in Table 5.1.4. (You can use the options in Table 5.1.4 without any of the flags shown in Table 5.1.3—that is, to get final results—but they are generally most useful for judging convergence, particularly of geometry optimizations.)

## *Reporting Results for Each Atom*

As described above, by default, each line of output from a "jaguar results" command lists information that pertains to the entire input structure, but you can also request some kinds of information for each individual atom in the structure.

Table 5.1.5 lists the "jaguar results" options that let you print tables of coordinates, forces, or charges for individual atoms. You can use these options in combination with the options from Table 5.1.1, which describe the job run, or with the options from Table 5.1.3, which

***Table 5.1.4.*** *"jaguar results" options that are primarily useful for reporting information on intermediate stages of a job (for example, steps in a geometry optimization).*

| jaguar results flag | corresponding jaguar results row output |
|---|---|
| iterg | geopt iteration number |
| stepg | geopt step number |
| grms | rms gradient |
| gmax | maximum gradient component |
| drms | rms displacement |
| dmax | maximum displacement |
| echange | energy change |
| iter | number of scf iterations (cumulative) |

***Table 5.1.5.*** *"jaguar results" options that let you view results for individual atoms. The "-atoms" and/or "-atomnums" flag request results for each atom, and the other flags dictate the type of results reported.*

| jaguar results flag | corresponding jaguar results row output |
|---|---|
| atoms | atom labels |
| atomnums | atom numbers |
| coords | Cartesian atomic coordinates |
| forces | Cartesian atomic forces |
| charges | ESP atom-centered charges |

request output for different stages of a job. However, you should not use the atom-related "jaguar results" with any of the options that request information pertaining to the entire molecule (the "-energy" option, for instance).

# 5.2. OUTPUT FROM A STANDARD HF CALCULATION

The contents of a Jaguar output file vary according to the calculation and output selections made. This section describes the output file for a standard, default, single point, closed shell Hartree-Fock calculation. Section 5.3 describes the variations in the output file for the calculation options described in Chapter 3.

All output files begin with a line listing the job name, the machine upon which the job ran, and the time the job was started, followed by the general copyright information for the version of Jaguar which was used for the run. The rest of this section describes output from each individual Jaguar program run for a default calculation.

The output from the program pre begins with a description of the calculation to be performed: its job name, the directory containing the executables used to run the job, the directory containing the temporary files, comments from the input file (if any), and the names and paths of any non-default data files used for the calculation (as explained in section 8.2 and Chapter 9). Comments from the input file include any text entered in the Comment box in the Run or Save window, as well as a comment about the point group if the geometry was symmetrized as described in section .

Next, the basis set used for the calculation, the molecule's net charge and multiplicity, and the number of basis functions used for the calculation are specified. This information is followed by the molecular geometry input, which gives the atom label and coordinates for each atom. (If the atom labels provided in the geometry are not unique—for instance, if two hydrogens are each called 'h'—this information is preceded by a list of original atom labels and new atom labels assigned by the program.)

The molecule's symmetry is analyzed, a process which may involve translating and rotating the molecule. These procedures are noted in the output file, along with the point group used for the calculation, the nuclear repulsion energy, and the symmetrized geometry, which is used for the rest of the calculation.

One-electron integrals are calculated by the onee program, which outputs the smallest eigenvalue of the overlap matrix **S** and the number of canonical orbitals used for the calculation. Canonical orbital eigenvectors with very small eigenvalues (less than $5.0 \times 10^{-4}$) are removed and thus are not counted. The eigenvalue cutoff can be controlled by setting the keywords **cut10** and **cut20** to the desired value in the **gen** section of the input file.

The program hfig constructs a starting wavefunction (initial guess) for a Hartree-Fock calculation. The output from the program hfig for a default calculation begins with the line, "initial wavefunction generated automatically from atomic wavefunctions." Next, a table lists the number of orbitals, and of occupied orbitals in each shell, having each irreducible representation for the appropriate point group. Finally, the orbital occupation for each shell is listed; an occupation of '1.000'

indicates a closed shell. An example, for a calculation of water using a 6-31G** basis set, follows:

```
 start of program hfig
 initial wavefunction generated automatically from atomic wavefunctions

 Irreducible     Total no   No of occupied orbitals
 representation  orbitals   Shell_1  Shell_2    ...
 A1                12          3
 A2                 2          0
 B1                 4          1
 B2                 7          1
 ------------------------
 Orbital occupation/shell    1.000

 end of program hfig
```

The probe program, which follows hfig and insures orthogonalization, has no significant output.

The output for the grid generation done by the program grid lists the number of grid points for each atom, as well as the total number of grid points, for each grid used in the application of the pseudospectral method. If you would like more information about these grids, see section 9.4. The rwr program, which generates the **Q** operators needed for the pseudospectral method, runs next, but has no significant output.

An example of the next output, from the program scf, is given here, again for a water molecule, and will be explained below.

```
 start of program scf
number of electrons.........         10
number of alpha electrons....         5
number of beta electrons.....         5
number of orbitals, total....        25
number of core orbitals......         5
number of open shell orbs....         0
number of occupied orbitals..         5
number of virtual orbitals...        20
number of hamiltonians.......         1
number of shells.............         1
SCF type: HF

      i  u  d  i  g                                          RMS    maximum
      t  p  i  c  r                                 energy  density   DIIS
      e  d  i  u  i                   total energy  change  change    error

etot  1  N  N  5  M   -75.61350567257                      1.6E-02  3.3E-01
etot  2  Y  Y  6  M   -75.99456008691  3.8E-01  6.2E-03  6.9E-02
etot  3  Y  Y  6  M   -76.01904109359  2.4E-02  1.7E-03  2.9E-02
etot  4  N  Y  2  U   -76.02333233097  4.3E-03  7.6E-04  4.7E-03
```

```
etot   5  Y  Y  6  M   -76.02361760760  2.9E-04  1.7E-04  1.5E-03
etot   6  Y  N  6  M   -76.02364072535  2.3E-05  0.0E+00  0.0E-00


 Energy components, in hartrees:
   (A)  Nuclear repulsion............     9.33000672144
   (E)  Total one-electron terms.....  -123.34165776264
   (I)  Total two-electron terms.....    37.98801031585
   (L)  Electronic energy............   -85.35364744679  (E+I)
   (N)  Total energy.................   -76.02364072535  (A+L)

 SCFE: SCF energy: HF     -76.02364072535 hartrees   iterations:   6


 HOMO energy:    -0.49745
 LUMO energy:     0.21516

  Orbital energies/symmetry label:
 -20.55693 A1        -1.34635 A1        -0.71380 B2        -0.56828 A1
  -0.49745 B1         0.21516 A1         0.30862 B2         1.01720 B2
   1.09266 A1         1.13459 A1         1.16904 B1         1.29575 B2
   1.41126 A1         1.80256 A2         1.82999 A1

  end of program scf
```

The output from the program scf begins with a list of information detailing the number of electrons in the molecule, the number of alpha and beta electrons, the total number of orbitals for the calculation, the numbers of core, open shell, occupied, and virtual orbitals, the number of Hamiltonians used for the calculation, the number of shells, and the calculation type.

Next, the energy output from the SCF iterations is shown in table form. Some of the text for the column headings should be read down rather than across. The number of the iteration is provided first in each row, followed by a "Y" or "N" indicating whether the Fock matrix was updated or not. When the Fock matrix is updated, the changes are made using a differential density matrix whose elements simply reflect the changes in the density matrix elements from the previous iteration to the current one.

The next entry indicates whether the DIIS convergence scheme was used for that iteration. As above, "Y" or "N" indicate yes or no. The DIIS method produces a new estimate of the Fock matrix as a linear combination of previous Fock matrices, including the one calculated during that iteration. DIIS, which is enabled by default, usually starts on the second iteration, and is not used on the final iteration. If the entry in this column reads "A," it indicates that DIIS was not used for that iteration, but the density matrix was averaged.

The cutoff set for each iteration is indicated under the "icut" heading. Cutoff sets are explained in the cutoff file description in section 9.5.

The grid column lists the grid used for that iteration, which must be one of the grid types coarse (signified by a C), medium (M), fine (F), or ultrafine (U). See the subsection *Grid and Dealiasing Function Keywords* in section 8.6, and the description of the grid file in section 9.4, if you want more information on grids and grid types.

The total energy for the molecule in Hartrees appears in the next column, followed by the energy change, which is the difference in energy from the previous iteration to the current one.

The RMS density change column provides the root mean square of the change in density matrix elements from the previous iteration to the current one.

In the last column, the maximum DIIS errors listed provide a measure of convergence by listing the maximum element of the DIIS error vector. For HF and DFT closed shell calculations, the DIIS error vector is given by **FDS – SDF** in atomic orbital space, where **F**, **D**, and **S** are the Fock, density, and overlap matrices, respectively. For open shell and GVB cases, the definition of the error vector is given in reference [11].

After the energy information for each SCF iteration is provided, a summary of the components of the final, converged energy is listed. The nuclear repulsion, one-electron, two-electron, and electronic contributions are all listed, followed by the total. Each of these energies is labeled with a letter (for example, 'A' for the nuclear repulsion), and information to the right of some of the energies describes the relations between the components in terms of these letters. A line beneath the table summarizes the calculation type and energy, as well as the number of SCF iterations.

(If the input system's spin multiplicity is not singlet, the default, the summary of the SCF output also includes a breakdown of the two-electron contribution to the energy into Coulomb and exchange parts. For each of these parts, the contribution from each Hamiltonian is listed.)

The highest occupied molecular orbital (HOMO) and lowest unoccupied molecular orbital (LUMO) energies are listed next. Finally, the energies for each occupied orbital and for the ten lowest-energy virtual orbitals (if that many exist) are provided, with each orbital identified by a symmetry label. Virtual orbitals and eigenvalues are determined in the same manner as in ref. [89]. The virtual orbitals are

obtained by diagonalizing $H_0 + \sum f(2J - K)$, where $f$ is the occupation of each orbital (1 for a closed shell). (For closed shell Hartree-Fock calculations, this definition yields the standard orbitals and eigenvalues.)

Finally, the cpu time for the job, the machine upon which the job ran, and its time of completion are noted at the end of the output file.

# 5.3. OUTPUT FILE CHANGES FOR CALCULATION OPTIONS

Any time you make a non-default setting for a calculation, the output from the program pre will note the non-default options chosen. This output will appear above the molecular geometry output from the pre program. This section describes the changes in output for various calculation settings described in Chapter 3.

Generally, only the *format* changes that result from these settings are discussed below! Naturally, these settings will often change the data listed. For information on the settings themselves, see Chapter 3. Options which have no significant impact on the output *format* are not discussed in this section.

## *DFT*

If you use density functional theory for the SCF calculation, the output above the SCF table lists the functional or combination of functionals used. The energy information for DFT calculations includes the breakdown of the two-electron energy into Coulomb and exchange-correlation terms. For DFT calculations, virtual orbitals are obtained by diagonalizing $H_0 + \sum f(2J + V_{xc})$, where $f$ is the occupation of each orbital (1 for a closed shell). (For closed shell calculations, this definition yields the standard orbitals and eigenvalues.)

The scf output from post-SCF DFT energy evaluations (GVB-DFT calculations, for instance) first lists the standard output for the HF, GVB, or DFT SCF calculation, then lists the energy breakdown and total energy from the post-SCF DFT analysis. Since the post-SCF DFT treatment does not change the wavefunction, no orbital output is reported from this step.

The output from the program pre for non-default options contains the detailed description of customized functional combinations for SCF or post-SCF DFT calculations.

## *LMP2*

If you perform a local MP2 calculation, the output from the programs pre and hfig is somewhat different than for a Hartree-Fock calculation, since the use of symmetry is turned off automatically for LMP2 calculations. The output from the program scf includes the Coulomb and exchange contributions to the two-electron terms for these calculations, and the symmetry labels are not included in the output of orbital energies.

The program loclmp2, which computes localized orbitals, runs after scf in an LMP2 calculation, and its output notes the number of orbitals that are localized. Below that output, the output from the program lmp2 appears.

For local local MP2 calculations, the output begins by listing the localized orbitals involved in the local local MP2 treatment—namely, the localized orbitals centered on one or both atoms in the pairs of atoms for which an LMP2-level treatment was requested.

All LMP2 output includes a description of the type of orbitals used in the MP2 calculation. First, it lists the total number of orbitals. Next, it lists the number of frozen core and valence MP2 orbitals. The numbers of core and valence orbitals will be affected by your choice from the LMP2 window of whether to use valence electrons only or all electrons for the atoms in the calculation. Next, the numbers of occupied and virtual orbitals for the molecule are listed. The list ends with the number of exchange Hamiltonians.

Some information on the convergence of the LMP2 energy correction appears below the list of orbital information, followed by the Hartree-Fock energy and the LMP2 energy correction, which gives the improvement to the energy over the HF value. The total LMP2 energy (the HF energy plus the correction) is given immediately afterwards. (If your job is a local local MP2 calculation and you want to see the energy from each LMP2 pair, use the **gen** section keyword setting **ip170**=2, as described in section 8.6 under *Standard Output Keywords*.)

## *GVB*

If a GVB calculation is performed from a Hartree-Fock initial guess, the pre program output lists a table of GVB pair information below the list of non-default options. The information in the table includes whether a restricted configuration interaction (RCI) calculation including that pair will be performed (Y or N for yes or no), and what

the configuration interaction (CI) coefficients are for the pair. Since the use of symmetry is turned off automatically for GVB calculations, the output from the programs pre and hfig is somewhat different than for a Hartree-Fock calculation. Also, the program gvbig runs after hfig, if the GVB initial guess is being generated from the HF initial guess.

The output from the scf program is more extensive than for a default HF calculation. First, the number of GVB pairs and the number of GVB orbitals are added to the list of electron and orbital information preceding the table of SCF iteration information. Secondly, the summary of the SCF output is followed by a breakdown of the two-electron contribution to the energy into Coulomb and exchange parts. For each of these parts, the contribution from each GVB Hamiltonian is listed. After this information, the intra-pair exchange energies and their sum are listed. Finally, a table of GVB pair information is given. Here is an example of this GVB information in the SCF output, for a water molecule with two GVB sigma pairs:

```
                                 Total           Coulomb          Exchange
 Total two-electron terms   37.90378136033   46.96140169504   -9.05762033471
   Hamiltonian  1........   25.77166631229   32.84704880440   -7.07538249211
   Hamiltonian  2........    6.02807668738    6.99023521309   -0.96215852571
   Hamiltonian  3........    6.02990515066    6.99271668375   -0.96281153309
   Hamiltonian  4........    0.03711925295    0.06576591758   -0.02864666463
   Hamiltonian  5........    0.03701395705    0.06563507622   -0.02862111917

  List of Intra-Pair K Energies
    -0.03983705429    -0.03981442075
    Sum of Intra-Pair K Energy...    -0.07965147505


 GVB pair information:

      first natural orbital  second natural orbital
     ----------------------  ------------------------              ci energy
pair orb ham shl  ci coeff.  orb ham shl  ci coeff.    overlap    lowering
---- --- --- --- ----------- --- --- --- ----------- ----------- -----------
  1   4   2   2  0.995433818  6   4   4 -0.095454256 0.824997160 0.020103338
  2   5   3   3  0.995443725  7   5   5 -0.095350881 0.825171705 0.020091467

SCFE: SCF energy: GVB     -76.06328826029 hartrees   iterations:   8
```

Each row in the GVB pair information table lists the pair number, the orbital number (after all core and open orbitals have been assigned numbers), Hamiltonian number (after the core Hamiltonian and any open Hamiltonians have been assigned numbers), and shell number (after the core shell and open shell, if they exist, have each been assigned a number) corresponding to each natural orbital, and CI coefficient corresponding to each GVB natural orbital in the pair.

Next, the overlap between the two corresponding non-orthogonal orbitals for that pair is listed, followed by the CI energy lowering, which is a guide to the energy change resulting from the inclusion of the second natural orbital in the calculation.

If a GVB calculation is performed from a Hartree-Fock converged wavefunction, the program scf runs twice, once to obtain the HF converged wavefunction, and once to perform the final GVB calculation. The SCF output from the first scf run will look like the scf output from a standard HF calculation; the output from the second run will have the format described above for a GVB calculation from an HF initial guess.

## *GVB-RCI*

For restricted configuration interaction calculations, the SCF output is the same as for non-RCI GVB calculations, but the output from the program rci appears after the SCF output. The RCI output first lists information on the total number of orbitals, the number of core orbitals for the RCI calculation, the numbers of open shell and GVB orbitals, the number of GVB and RCI pairs, the numbers of occupied and virtual orbitals, the numbers of Coulomb and exchange Hamiltonians, and the multiplicity. Next, the total energy of the GVB wavefunction, which was obtained from the SCF procedure earlier, is listed and broken down into a nuclear repulsion term and terms from the core, which is treated with Hartree-Fock methods, the GVB pairs, and the open shell contribution. If you specified GVB pairs that were not also RCI pairs, a non-zero value is listed for the non-RCI GVB pair energy. The number of RCI spatial configurations and the number of RCI configuration state functions follow. Each RCI configuration state function is the product of a contracted spatial function and a spin function; the number of these functions indicates the size of the RCI expansion.

The rci output lists the RCI initial guess at the energy next, followed by the output from the calculation of the converged total RCI energy. For small cases, the initial guess and the converged energy may agree exactly, since the RCI coefficients are obtained by one diagonalization of a small matrix. For larger cases, the output includes the results of the iterative diagonalization.

## *Geometry or Transition State Optimization (HF, GVB, DFT, and LMP2)*

The output format description for optimizations in this subsection applies to calculations of either minimum-energy structures or transi-

tion states. Although the Hessians used during these calculations are different, the Jaguar programs run are the same, and the output format is very similar. (Exceptions are described below.)

If you calculate an optimized molecular structure, a transition state, or forces, any SCF calculations during the run use the RMS density change convergence criterion described in section 3.9 instead of the usual energy convergence criterion. Therefore, these SCF calculations often proceed for several more iterations than single point energy calculations yield.

If you select forces only for the Optimize geometry setting, the programs der1a, rwr, and der1b will run after scf does. The forces felt by each atom in the unoptimized geometry will be output from der1b, in a table listing each atom and the components of the force upon it in the x, y, and z directions. The x, y, and z components of the total force on the molecule are listed in the last line, and provide a judge of how accurate the force calculations are in most cases, since they should generally be zero. An example of this force table for a water molecule optimization follows:

```
 forces (hartrees/bohr) : total

atom  label          x             y             z
----  -----   -------------  -------------  -------------
  1     O       0.000000E+00   0.000000E+00  -2.620407E-05
  2     H1      0.000000E+00  -6.462331E-05   1.291533E-04
  3     H2      0.000000E+00   6.462331E-05   1.291533E-04
-----------   -------------  -------------  -------------
   total        0.000000E+00   0.000000E+00   2.321025E-04
```

When force calculations or optimizations of a system's minimum energy structure or transition state are performed at the LMP2 level, the program der1b never runs. Instead, forces are calculated by the programs lmp2der, lmp2gda, and lmp2gdb. The last of these programs provides a table of output listing the forces on each atom in the same format as the sample table above.

If Optimize geometry is set to minimum energy or transition state, Jaguar prints bond length and angle information in the output from the program pre. This information appears in the output file because the bond lengths and angles option from the Standard window under the Output heading is automatically enabled for geometry optimizations. If you have constrained certain bond lengths or angles of the geometry so that they are frozen during the optimization, as described in section 4.2, the constraints are also listed in the pre output.

At the end of the first SCF calculation, the programs der1a, rwr, and der1b run, calculating the forces felt by each atom in the unoptimized geometry and writing them to the output file, as described above.

These force results are followed by the output from the program geopt, which includes a number indicating how many times it has been called (for example, 1 for the first iteration) in the "start of program geopt" line. Every time geopt is called, this number is updated. However, since geopt can be called for Hessian refinement steps as well as for generating new geometries during an optimization, and since geometry optimizations occasionally revert back to a previous geometry and "restart" the calculation from there, the next line of the geopt output reports what sort of step is being performed and numbers that step accordingly.

If the program detects that the input lists separate fragments, each of which contain only atoms unbonded to the atoms in any other fragment, as for a van der Waals complex, then the number of fragments is listed near the start of the geopt output.

For transition state optimizations, the eigenvalues of the nuclear Hessian are reported the first time geopt runs. If the initial Hessian is being refined, the coordinates for the refinement and their eigenvalues are listed. (If a coordinate you have specified is inappropriate because of symmetry restrictions or other constraints, the output will indicate the problem.) The geopt output then lists information on the current (original) geometry's gradient elements, describes the small step it will use to alter the first coordinate used in the Hessian refinement, describes the internal coordinates and optimization variables as stretches, bends, or torsions, and indicates how it generates a new geometry by altering the relevant coordinate by the amount described by the step size.

The new geometry generated for Hessian refinement is then used to obtain energy and gradient information, a process that requires the programs onee, grid, and rwr to run and generate output in the usual formats, which is followed by output from the program scf, which now starts with the calculation type and the table showing the energy output from each SCF iteration (skipping the listed information about electrons, orbitals, and so on). The output further continues with output in the usual formats from der1a, rwr, and der1b. The information obtained on that geometry is then used in geopt, which runs a second time, reporting similar information about the planned changes to the molecular structure for the next Hessian refinement step (if there is one) and reporting the change in total energy from the original geometry to the geometry for the first Hessian refinement step as

well. This process of altering single coordinates from the original geometry and calculating energies and gradients for the changed geometry continues until all requested Hessian refinement steps have been performed, which the output indicates with a line beginning "Hessian optimization completed." After that point, geopt performs a geometry optimization step from the original geometry, and the actual optimization continues until convergence.

For transition state optimizations, the output for iterations that follow any Hessian refinement includes information identifying the transition vector used for that iteration. This output includes the transition vector's eigenvalue and the stretches, bends, or torsions that are its most important components.

For any optimization iteration using level shifting, after any relevant lines of geopt output described above, some information on the computed level shift (which may then be adjusted to satisfy step-size constraints) is included in the output. For optimization steps past the first geometry change, the change in total energy from the previous geometry to the newly calculated geometry (in Hartrees) is listed next.

The geopt output then notes the maximum element of the analytic gradient calculated by the earlier programs; the root mean square of the gradient elements; the step size predicted for the geometry change, the trust radius for that iteration and, if it is smaller than the step size, the factor used to scale the step size so it is no larger than the trust radius; the maximum element of nuclear displacement; and the root mean square element of the nuclear displacement. The predicted energy change for the new structure generated by geopt is also listed.

The values for the energy change, gradient, and nuclear displacement described in the previous paragraph are important because they are each tested against the convergence criteria determined by the Convergence criteria setting from the interface's Optimization window, as described in section 4.1, or, alternatively, the criteria set by the **gconv** keywords in the input file. The criteria are described in detail in the *Geometry Optimization and Transition State Keywords* subsection of section 8.6. If the gradients are converged and the energy change is below $2.5 \times 10^{-7}$, the optimization will stop (unless it is on the first geometry optimization iteration). Similarly, if the gradients are converged and one of the gradient criteria is 5 times lower than the convergence level, then the optimization will stop if the energy change is less than $2.5 \times 10^{-6}$.

The symbol following each quantity used to judge convergence indicates how well converged it is. The symbol '.' indicates convergence criteria that are not satisfied, '*' indicates criteria that are satisfied, '#' indicates criteria that are quite well satisfied, '!' indicates values that are essentially zero. If the convergence criteria mentioned are not sufficiently well met, and if the geometry optimization has not already gone through as many iterations as the Maximum iterations setting from the Optimization window, the output will note "molecular structure not yet converged..." and the optimization will continue.

The output next lists the movement of the center of mass. If the output option for the bond length and angles is enabled, the output then lists this information for the new structure. Finally, the nuclear repulsion energy for the new geometry is listed.

If the molecular structure was not yet converged and the maximum number of geometry optimization iterations allowed was not reached in the previous iteration, the output from more geometry optimization iterations will follow. The output from each iteration begins with onee, grid, and rwr output in the usual formats, and continues with output from scf, which now starts with the calculation type and the table showing the energy output from each SCF iteration (skipping the listed information about electrons, orbitals, and so on). The output further continues with output in the usual formats from der1a, rwr, and der1b, winding up with the output from geopt. The last such geometry optimization iteration contains, in the geopt output, either the line, "Geometry optimization complete," or the line, "stopping optimization: maximum number of iterations reached," depending on whether the convergence criteria were met before the maximum number of iterations was reached.

## *Optimizations with GVB-RCI Wavefunctions*

Geometry or transition state optimizations using GVB-RCI run in much the same way as described above for HF, GVB, or DFT optimizations, except that the forces for the optimization are computed numerically rather than analytically. Consequently, the der1a and der1b programs never run; instead, when forces are needed, the structure's energy is evaluated at $6N_{atom}$ perturbed geometries, where $N_{atom}$ is the number of atoms, and the forces are computed numerically. The program nude generates each perturbed geometry by moving an atom a small amount in the positive or negative x, y, or z direction, and also evaluates the numerical derivatives when calculations on all perturbed geometries are complete, listing them in the output in a force table similar to the usual geometry optimization force table described for HF, GVB, or DFT systems. The program geopt still runs in the usual

way as well, computing each iteration's new geometry using the available forces.

## Solvation

Performing a solvation calculation involves several iterations in which the wavefunctions for the molecule in the gas phase are calculated; the program ch performs electrostatic potential fitting which represents the wavefunction as a set of point charges on the atomic centers; the interactions between the molecule and the solvent are evaluated by Jaguar's Poisson-Boltzmann solver, which fits the field produced by the solvent dielectric continuum to another set of point charges; these solvent point charges are passed back to scf, which performs a new calculation of the wavefunction for the molecule in the field produced by the solvent point charges; ESP fitting is performed on the new wavefunction; the solvent-molecule interactions are reevaluated by the Poisson-Boltzmann solver; and so on, until the solvation energy for the molecule in the solvent converges.

For solvation calculations on neutrally charged systems in water whose atoms all have atomic numbers under 19 (H-Ar), by default, the program pre evaluates the Lewis dot structure for the molecule or system and assigns atomic van der Waals radii accordingly. (For more information on this process, see section 9.6.) These van der Waals radii are used to form the boundary between the solvent dielectric continuum and the solute molecule. The Lewis dot structure and van der Waals radii information both appear in the output from the program pre. The radii are listed under the heading "vdw2" in the table of atomic information below the listing of non-default options. See section 8.9, which describes the **atomic** section of the input file, if you want information on the other information in this table.

After the pre output, the usual output appears for the first, gas-phase calculation, except that the energy breakdown for the scf output also describes the electron-nuclear and kinetic contributions to the total one-electron terms in the energy, as well as the term -V/T (where V is the potential energy and T is the kinetic energy), which indicates how well the calculation is agreeing with the virial theorem (it should be -2.00).

After the first scf output, the output from the first run of the program ch appears. Since performing a solvation calculation enables ESP fitting to atomic centers, the usual output for that option, which is described in the subsection *Properties* later in this section, is included every time output from the program ch appears in the output file. The

post program writes out the necessary input files for the Poisson-Boltzmann solver; this step is noted in the output file.

The next output section comes from the Poisson-Boltzmann solver. The output includes information on the area (in Å$^2$) of the molecular surface formed from the intersection of spheres with the van der Waals radii centered on the various atoms; the reaction field energy in kT (where T = 298 K), which is the energy of the interaction of the atom-centered charges with the solvent; the solvent-accessible surface area (in Å$^2$), which reflects the surface formed from the points whose closest distance from the molecular surface is equal to the probe radius of the solvent; and the cavity energy in kT, which is computed to be the solvation energy of a nonpolar solute whose size and shape are the same as those of the actual solute molecule, as described in reference [15].

The output from the Jaguar program solv follows the Poisson-Boltzmann solver results, giving the number of point charges provided by the solver to model the solvent, the sum of the surface charges, the nuclear repulsion energy already calculated by Jaguar, the nuclear-point charge energy representing the energy of interaction between the molecule's nuclei and the solvent point charges, and the point-charge repulsion energy, which is calculated but not used by the rest of Jaguar because it is irrelevant to the desired solvation results.

After this output, the output for the second solvation iteration begins. The output from scf comes first, giving the results for the molecule-and-solvent-point-charges system. An example, from the first solute-with-solvent-point-charges scf run in a calculation of 6-31G** water in cyclohexane, using the Jaguar solver, is given here:

```
 start of program scf

      i  u  d  i  g
      t  p  i  c  r                                    RMS    maximum
      e  d  i  u  i                        energy   density   DIIS
      r  t  s  t  d     total energy       change   change    error

etot  1  N  N  2  U   -76.03588607997              6.8E-04  6.6E-03
etot  2  Y  Y  6  M   -76.03615425936  2.7E-04     1.9E-04  1.8E-03
etot  3  Y  N  6  M   -76.03617415619  2.0E-05     0.0E+00  0.0E+00


Energy components, in hartrees:
  (A)  Total zero-electron terms....     9.35161183359
  (B)    Nuclear-nuclear............     9.33000672144
  (C)    Nuclear-solvent............     0.02160511215
  (E)  Total one-electron terms.....  -123.39806065860
  (F)    Electron-nuclear...........  -199.21812919134
  (G)    Electron-solvent...........    -0.03443064237
  (H)    Kinetic....................    75.85449917511
```

```
  (I)  Total two-electron terms.....    38.01027466882
  (L)  Electronic energy............   -85.38778598978  (E+I)
  (N)  Total quantum mech. energy...   -76.03617415619  (A+L)
  (O)  Gas phase energy.............   -76.02364072535
  (P)  Solution phase energy........   -76.02607108661  (Q+R+S)
  (Q)    Total solute energy........   -76.02334862596  (N-C-G)
  (R)    Total solvent energy.......    -0.00641276511  (C/2+G/2)
  (S)    Solute cavity energy.......     0.00369030447
  (U)  Reorganization energy........     0.00029209939  (Q-O)
  (V)  Solvation energy.............    -0.00243036126  (P-O)

SCFE: SCF energy: HF     -76.03617415619 hartrees   iterations:   3


HOMO energy:    -0.49985
LUMO energy:     0.22469

 Orbital energies/symmetry label:
 -20.55803 A1        -1.34624 A1        -0.71287 B2        -0.57176 A1
  -0.49985 B1         0.22469 A1         0.31901 B2         1.01892 B2
   1.09275 A1         1.13045 A1         1.16509 B1         1.29393 B2
   1.41452 A1         1.80375 A2         1.82851 A1

 end of program scf
```

As for any later solvation iterations, the scf output begins with the calculation type and the table of energy results for each iteration, skipping the list of information about the molecule's electrons and orbitals, and the energy information below the table includes several additional terms, whose relations to each other are described with the usual alphabetic labels. First, the total of the terms with no electron contribution is listed (term (A)), followed by terms (B) and (C), the nuclear-nuclear and nuclear-solvent energies.

Next, the total one-electron energy is listed, along with its three components, the electron-nuclear, electron-solvent, and kinetic energies. The total two-electron energy, and the total of the one- and two-electron energy terms, the electronic energy, follow. Term (N), the total of the zero-, one-, and two-electron terms, is then listed, with the label "Total quantum mech. energy." This term corresponds to the final energy from the scf energy table for that iteration, and includes the entire energies for the molecule-solvent interactions.

The output next includes the gas phase and the solution phase energies for the molecule, since these terms are, of course, necessary for solvation energy calculations. The first solution phase energy component is the total solute energy, which includes the nuclear-nuclear, electron-nuclear, kinetic, and two-electron terms, but no terms involving the solvent directly. The second component of the solution phase energy is the total solvent energy, which is computed as half of the total of the nuclear-solvent and electron-solvent terms, since some

of its effect has already changed the solute energy. Third, a solute cavity term, which computes the solvation energy of a nonpolar solute of identical size and shape to the actual solute molecule, as described in reference [15], is included. The last solution phase energy component (shown only if it is nonzero) is term (T), the first shell correction factor, which depends on the functional groups in the molecule, with atoms near the surface contributing most heavily.

Finally, the list ends with the reorganization energy and the solvation energy. The reorganization energy is the difference between the total solute energy and the gas phase energy, and does not explicitly contain solvent terms. The final solvation energy is calculated as the solution phase energy described above minus the gas phase energy.

The results of the self-consistent reaction field iterations so far performed are summarized after the scf output in the output from the program sole. An example from the final SCRF iteration of water in cyclohexane follows:

```
start of program sole
  SCRF                    solvation energy
iteration          Hartrees            kcal/mol
   0               0.0000000            0.0000
   1              -0.0024304           -1.5251
   2              -0.0027473           -1.7240
   3              -0.0027918           -1.7519

stopping: solvation energy converged

iterations:   3    sfinal:     -1.7519 kcal/mol

end of program sole
```

The solvation energy is listed in Hartrees and in kcal/mole, and the note beneath it reads either "solvation energy not yet converged..." or "stopping: solvation energy converged," depending on whether the solvation energy has changed by less than the Solvation convergence criterion, which is described in section 3.5. If the solvation energy has converged, the output from the sole program includes a line summarizing the solvation energy iterations and result.

The output from ch and post appears below the sole output. If the solvation energy has converged, the ch output reflects the system's final atomic charges. If the solvation energy has not converged, these charges and the Poisson-Boltzmann solver's files generated by the post program are passed to the solver again, and the solvation iterations continue as previously described, until solvation energy convergence is reached.

## *Geometry Optimization in Solution*

Geometry optimizations in solution contain output in the formats described in the previous two subsections, but the optimization output and the solvation calculation output alternates as the calculation proceeds. First, by default, Jaguar computes a gas phase optimized geometry, for which the output is the same as that described above for a standard optimization. Next, the SCRF procedure is used to compute a wavefunction for the solvated system, as for a single point solvation energy calculation. When the solvation energy has converged, Jaguar runs the program pbf once more to get the solvation-related gradient. This pbf output does not contain the usual solvent accessible surface area and cavity energy terms. The programs der1a, dsolv, rwr, and der1b then compute the forces, with the force table in the der1b output in the usual manner, and the program geopt computes the new molecular structure, as usual. For each new structure generated during the optimization, Jaguar first performs the SCRF calculation, then obtains the forces (in solution), and finally generates a new structure. The calculation proceeds until the geometry optimization convergence criteria are reached. The convergence criteria for optimizations in solution are three times larger than they are for optimizations in the gas phase.

For solvated geometry optimizations, the solvation energy is computed as the difference between the energy of the optimized gas phase structure and the energy of the solvated structure that was optimized in solution.

## *Properties*

If you make any non-default selections from the Properties window, the program ch runs and outputs the results to the output file after the SCF iterations, if any.

When multipole moments are calculated, the x-, y-, and z-direction components of the dipole moment and the total magnitude of the dipole moment are reported in Debye, followed by information on any requested higher-order moments and the corresponding traceless higher-order moment tensors. For example, here is the output for a calculation of water's dipole and quadrupole moments:

```
Moments from quantum mechanical wavefunction:
Dipole Moments  (Debye)
   X=     0.0000     Y=     2.1470     Z=     0.0000  Tot=     2.1470
```

```
 Quadrupole Moments  (Debye-Ang)
   XX=    -4.0828    YY=    -5.7670    ZZ=    -7.1340
   XY=     0.0000    XZ=     0.0000    YZ=     0.0000
 Traceless Quadrupole Moments  (Debye-Ang)
XX-YY=     1.6843               2ZZ-XX-YY=    -4.4182
   XY=     0.0000    XZ=     0.0000    YZ=     0.0000
```

If ESP charge fitting to atomic centers is performed, the output lists the number of grid points from the charge grid, which is used for the charge fit. It then describes the constraint or constraints for the fit, followed by the calculated atomic charges and their sum. The root mean square error of the charge fitting is also reported; this error is calculated from examining the Coulomb field at each grid point that would result from the fitted charges, and comparing it to the actual field.

If ESP fitting to atomic centers and bond midpoints is performed, the bond midpoints are treated as "dummy atoms" and their descriptions and coordinates are provided before the grid points information. The bond charges from the fit are provided, with the label "bond", along with those on the atomic centers. An example of the output from such a calculation follows, for water:

```
 dummy atom x4   is between            2 and            1
 dummy atom x5   is between            3 and            1

                                angstroms
 atom            x                 y                 z
 O         0.0000000000     -0.1135016000      0.0000000000
 H1        0.7531080000      0.4540064000      0.0000000000
 H2       -0.7531080000      0.4540064000      0.0000000000
 x4        0.3765540000      0.1702524000      0.0000000000
 x5       -0.3765540000      0.1702524000      0.0000000000

 gridpoints used for charge fit         4162
   out of a possible maximum of         4188

  Electrostatic potential fitting constrained to reproduce
    total charge:               yes
    dipole moment:              no
    traceless quadrupole moment: no
    traceless octapole moment:   no

 Atomic charges from electrostatic potential:

Atom       O        H1       H2       x4       x5
Charge   -0.31208  0.63681  0.63681 -0.48077 -0.48077

 sum of atomic charges:     0.000000

RMS Error   8.26E-04 hartrees
```

If the fit is constrained to reproduce the dipole moment (or dipole and higher moments), or any other time both ESP fitting and multipole moment calculations are performed, a new moment (or moments) can be calculated from the fitted charges, as described in section 3.6 in the subsection *Electrostatic Potential Fitting*. The output from ch begins with the moment or moments calculated for the quantum mechanical wavefunction, in the format for multipole moment calculations. Next, the electrostatic potential fitting information is provided, as described above. Finally, the components and totals of the moment or moments recalculated using the electrostatic potential charges are reported.

If you calculate polarizabilities and first hyperpolarizabilities with the coupled perturbed HF method, the tensor elements in A.U. appear in the output from the program cpolar, which runs after the SCF calculation. Alternatively, if you use the finite field method to calculate the polarizability and/or first hyperpolarizability of the molecule, the output includes data from all the SCF calculations involved. (See section 3.6 for details on the methods used to calculate polarizability and hyperpolarizability.) The data from the program scf includes the term -V/T (where V is the potential energy and T is the kinetic energy), which indicates how well the calculation is agreeing with the virial theorem (it should be -2.00). Before each SCF calculation used for the polarizability evaluation, the program polar runs and outputs the electric field (in A.U.) used for the SCF calculation whose output appears immediately afterwards. When all calculations needed for the finite difference method have been performed, the program polar outputs the polarizability tensor in A.U., the first hyperpolarizability tensor in A.U., if it has been calculated, and the dipoles from each SCF calculation, along with information about the electric fields used for the dipole calculations.

If you choose to calculate the electron density, the output from the program elden appears beneath the SCF output. The output lists the number of grid points used for the electron density calculation and the total number of electrons found over the grid. The main output file does not include the charges and grid points for the calculation; that information can be found in the output file *jobname*.chdens, where *jobname*.in is the input file for the Jaguar job. The file *jobname*.chdens lists the Cartesian coordinates and the electron density in A.U., respectively, for each grid point.

If you choose to calculate Mulliken populations by atom, the charge for each atom and the sum of the atomic charges will be noted under the heading "Atomic charges from Mulliken population analysis." If you choose to calculate them by basis function, the atomic charge output will be preceded by a section labeled "Mulliken population for

basis functions," listing the atom label, function (labeled with consecutive numbers), type of basis function (S for s, X for $p_x$, XX for $d_{xx}$, etc.), and calculated population. Calculating Mulliken populations by bond yields the populations by atom and basis function as well. An example of this output for a calculation of water using the 6-31G** basis set is provided below.

```
 Mulliken Bond Populations:  first nearest neighbor
 Atom1 Atom2  Pop.  Atom1 Atom2  Pop.  Atom1 Atom2  Pop.  Atom1 Atom2  Pop.
 H1    O     0.314  H2    O     0.314

 Mulliken Bond Populations:  second nearest neighbor
 Atom1 Atom2  Pop.  Atom1 Atom2  Pop.  Atom1 Atom2  Pop.  Atom1 Atom2  Pop.
 H2    H1   -0.025

 Mulliken population for basis functions
 atom  func.  type   population
 O       1       S     1.9954
 O       2       S     0.8942
 O       3       X     0.8034
 O       4       Y     0.9514
 O       5       Z     1.1426
 O       6       S     0.8865
 O       7       X     0.4669
 O       8       Y     0.6649
 O       9       Z     0.8332
 O      10      XX     0.0085
 O      11      YY     0.0024
 O      12      ZZ     0.0052
 O      13      XY     0.0142
 O      14      XZ     0.0000
 O      15      YZ     0.0021
 H1     16       S     0.4950
 H1     17       S     0.1263
 H1     18       X     0.0185
 H1     19       Y     0.0138
 H1     20       Z     0.0111
 H2     21       S     0.4950
 H2     22       S     0.1263
 H2     23       X     0.0185
 H2     24       Y     0.0138
 H2     25       Z     0.0111

 Atomic charges from Mulliken population analysis:

Atom       O        H1       H2
Charge   -0.67059  0.33530  0.33530

 sum of atomic charges:     0.000000
```

You may find it helpful to select the Gaussian function list (basis set) setting from the Standard window, whose button appears under the Output heading, if you wish to have more information about the basis functions. More information on this output option is given in Section 5.4.

If both Mulliken populations and multipole moments are calculated, the multipole moments are calculated from the atomic Mulliken populations as well as directly from the wavefunction, as noted in section 3.6 in the *Mulliken Population Analysis* subsection. The output lists the multipole moments from the wavefunction, as described earlier; the Mulliken populations, as described just above; and finally the recalculated moments resulting from the Mulliken charges, in the same format used for the earlier moment output.

If you perform an NBO calculation, its output will appear under the heading "Jaguar NBO 4.0."

## *Frequency, IR Intensity, and Thermochemistry Output*

If you calculate vibrational frequencies by making the appropriate setting in the Frequencies window, any SCF calculations during the run use the RMS density change convergence criterion described in section 3.9 instead of the usual energy convergence criterion. Therefore, these SCF calculations often proceed for several more iterations than single point energy calculations yield.

To compute vibrational frequencies from a Hessian computed during the same job, Jaguar sometimes calculates the second derivatives numerically as the derivatives of the analytical first derivatives, and other times computes analytic frequencies. (See section 3.7 for details.) Whenever numerical second derivatives are computed after an SCF calculation—whether for frequency output, for an initial Hessian, or for updating during geometry optimization—the programs nude, onee, hfig, grid, rwr, scf, der1a, rwr, and der1b run, setting up and performing SCF calculations and evaluating analytic gradients at $6N_{atom}$ perturbed geometries (unless the number of perturbed geometries needed is reduced by the use of molecular symmetry). To make each perturbed geometry, one atom is moved a small, fixed amount in the positive or negative direction along the x, y, or z Cartesian axes. After the calculations at the perturbed geometry, Jaguar performs one final calculation at the unperturbed geometry. (The Jaguar programs run may vary slightly for non-HF calculations, as described earlier in this section.) After the data from all perturbed geometries is collected, the program nude outputs the numerical first derivatives in a force table similar to the usual geometry optimization force table described earlier in this section. The output then lists the matrix indices of the most asymmetrical Hessian element before symmetrization. (The symmetrized numerical Hessian is not printed in the output, but can be found in the restart file, which is discussed in section 6.4.)

For either analytic or numerical frequency calculations, the output from the program freq contains the actual frequencies and normal modes from the computed Hessian, or from the last available Hessian (generally the initial Hessian guess) if you used the use available Hessian choice to request vibrational frequencies. The output from the program freq first lists the harmonic frequencies in $cm^{-1}$ and their symmetries (if symmetry is on for the job), then the normal modes. The system's thermochemical properties, the constant pressure heat capacity ($C_p$), entropy (S), enthalpy (H), and Gibbs free energy, are then listed for the specified pressure and temperatures, as well as at 0 K. Here is an example of this output from a vibrational frequency calculation on water:

```
start of program freq

harmonic frequencies in cm**-1, IR intensities in km/mol, and normal modes:

frequencies  1769.43  4148.13  4265.53
intensities   104.57    16.32    58.13
  O    X     0.00000  0.00000  0.06820
  O    Y     0.06810 -0.04857  0.00000
  O    Z     0.00000  0.00000  0.00000
  H1   X     0.40896  0.57347 -0.54119
  H1   Y    -0.54043  0.38540 -0.40782
  H1   Z     0.00000  0.00000  0.00000
  H2   X    -0.40896 -0.57347 -0.54119
  H2   Y    -0.54043  0.38540  0.40782
  H2   Z     0.00000  0.00000  0.00000


Thermochemical Properties:
  pressure:        1.0000 atm
  rotational symmetry number:    2
  zero point energy:     14.557 kcal/mol

   temp(K)   Cp (cal/mol K)   S (cal/mol K)   H (kcal/mol)   G (kcal/mol)
      0.00           7.9489         -0.3367         0.0000         0.0000
    298.15           7.9772         44.9564         2.3709       -11.0328

end of program freq
```

If infrared intensities were calculated, several additional programs will run after the first run of the program scf. These programs compute the derivatives of the dipole, which are needed to calculate the IR intensities. The IR intensities themselves are listed in the frequencies table described above, which appears in the freq output near the end of the output file.

### *Basis Set*

If your calculation uses a basis set that includes effective core potentials—that is, if the basis set's name begins with "LA"—the output lists the number of atoms treated with effective core potentials.

### *Methods*

If the DIIS convergence method is not used, the "maximum DIIS error" column is not printed for the table giving data from the SCF iterations. Also, if the OCBSE convergence scheme is selected, the Coulomb and exchange contributions to the total two-electron terms are listed in the SCF summary beneath the table.

If a fully analytic calculations performed, as happens for some basis set selections (see section 3.8 for more details), the programs grid and rwr will not run, because the all-analytic method cannot take advantage of pseudospectral speedups.

If you select a Final localization method, the output from the program locpost appears after the output from any SCF iterations and lists the orbitals that are localized. (If you want to print out the localized orbitals, you should make the appropriate selection in the Orbitals window, as described in Section 5.7.)

## 5.4. STANDARD OUTPUT OPTIONS

The menu options from the Standard window, whose button appears under the Output heading, are described in this section. The output generated from these options will appear in the output file for the job. If you make a non-default setting from the Standard window, the output from the program pre will note the non-default options chosen. This output will appear above the molecular geometry output from the same program, and will indicate the non-default values of the keywords referred to in footnotes throughout this section.

### *echo input file and parameter list*

If you turn this output option on, the output from the program pre will include an echo of the input file, a description of the path, which indicates the Jaguar programs run, and a list of keyword settings, including those made by default, and program parameters.[1] This

1. **echo** section constructed, and keywords **mtest** = 2 and **ip24** = 2 in **gen** section of input file.

option is likely to be useful primarily for people who have a detailed knowledge of the code itself.

## *memory, disk, and i/o information*

The memory information provided by this option is given for most of the routines used during the run, under the heading "dynamic memory statistics" each time.[2] Current and maximum values for the number of arrays, their size in 8 byte words, and their size in bytes, as well as the type of variables used (e.g. real*8), are listed. The total and index i/o for the J and K matrices, in Mwords, are also provided after the energy output from all of the individual SCF iterations.

## *detailed timing information*

If you select this option, the cpu seconds spent in various Jaguar programs will be listed in the output.[3]

## *geometries in atomic units also*

This option allows you to choose to print the geometry output in atomic units as well as in the usual units, Angstroms.[4]

## *multipole moments in atomic units also*

If you choose to calculate multipole moments by making the appropriate setting in the Properties window, this option allows you to choose to list them in the output file in atomic units as well as in the usual units, Debye.[5]

## *bond lengths and angles*

When this option is turned on, the internuclear distances in Angstroms are listed for all nearest neighbor atoms in the output from the program pre, and the bond angles in degrees are given as well.[6] The atoms are indicated with the atom labels assigned in the geometry input. When the Optimize geometry option in the Optimization window has been turned on, the bond lengths and angles standard output option is turned on automatically. For geometry optimizations,

2. Keyword **ip5** = 2 in **gen** section of input file.
3. Keyword **ip6** = 2 in **gen** section of input file.
4. Keyword **ip26** = 2 in **gen** section of input file.
5. Keyword **ip25** = 2 in **gen** section of input file.
6. Keyword **ip11** = 2 in **gen** section of input file.

bond lengths and angles are also listed with the output from the program geopt.

## *connectivity table*

The connectivity table provided by this option describes roughly how closely the atoms interact.[7] Connectivity partially determines whether molecular fragments exist, the content of the initial Hessian, and many other properties of a calculation. The assignment of dealiasing functions for the pseudospectral method also depends upon the connectivities shown in this table, which reflect the neighbor ranges defined in the .daf file. (See section 9.3 for more information.) All of the diagonal entries are 0, indicating that the row atom and the column atom for the matrix element are the same atom. An entry of 1 indicates that the row atom and the column atom are considered to be bonded, because they are separated by a distance less than the sum of their covalent radii times the variable **covfac**, which is 1.2 by default and is also described in the *Geometry Input Keywords* subsection of section 8.6. If a connectivity table entry is 2, the corresponding row and column atoms are each bonded to some same third atom, by the definition of bonding described above. An entry of 3, 4, or more means that the atoms are within the third, fourth, or higher neighbor range of each other.

## *overlap matrix*

The overlap matrix **S** for the basis functions will be printed in five-column blocks if this option is selected.[8] Since the matrix is symmetric, the elements within the top triangular half are not printed.

## *geometry optimization details*

If the geometry optimization details[9] option is selected, much additional information about the progress of a geometry optimization will be printed. This output often helps reveal the cause of any problems with optimizations.

## *one-electron Hamiltonian*

The one-electron matrices representing kinetic energy and the sum of kinetic energy, nuclear attraction, and point charge-electron interactions will be printed in atomic orbital space in five-column blocks if

7. Keyword **ip12** = 2 in **gen** section of input file.
8. Keyword **ip18** = 2 in **gen** section of input file.
9. Keyword **ip192** = 2 in **gen** section of input file.

this option is selected.[10] Since the matrices are symmetric, the elements within the top triangular halves are not printed.

## *Gaussian function list (basis set)*

By turning this option on, you can choose to print out information about the Gaussian functions that make up the basis set.[11] The orbitals in a basis set are made up of linear combinations of polynomials of the appropriate degree multiplied by Gaussian primitives of the form $Ne^{-zr^2}$, where N is a normalization constant and z is the exponent for the primitive. If the linear combination only includes one Gaussian primitive, the function is called uncontracted; otherwise, it is called a contracted Gaussian. The output controlled by this output option gives essentially the same information about the basis functions in two different tables, after giving a list of atoms and the basis set used for each one.

The shell information table is printed first. An example, for a calculation of water with a 6-31G** basis set, is given below. The first column of the table indicates which atom the primitive Gaussian being described in a particular row is centered upon. The second column lists the shell numbers, which increase consecutively for each atom. The values in the third column mean different things depending on their sign. The positive numbers mean that the basis function currently being described is composed of that number of primitive Gaussians, starting with the primitive Gaussian for that row and including the appropriate number of rows immediately beneath it. The negative numbers' magnitudes indicate the first shell which contributes to the same contracted Gaussian function. For instance, in the example below, the first row has a jcont value of 3, indicating that the first basis function being described is a contracted Gaussian composed of that primitive Gaussian and the two in the next two rows. The jcont values of -1 in the next two rows indicate that the primitive Gaussians being described are components in a contracted function whose first primitive Gaussian term is listed in the first row.

The values in the column marked "ishl" take on nonzero values when basis functions corresponding to different l values, as described in the next column, actually use primitive Gaussians with the same exponents. Positive values indicate that the same exponents should be used in the shell listed that number of rows down; a value of -1 indicates that the exponents should be provided from a shell listed earlier. The l values in the next column indicate angular momentum; a value of 1 corresponds to an s function, 2 indicates a p function, 3 a d function,

10.Keyword **ip19** = 2 in **gen** section of input file.
11.Keyword **ip1** = 2 in **gen** section of input file.

and so on. The nfsh values are one less than the lowest number corresponding to the basis function or functions being described. Note that the nfsh=2 entries below are for p functions, so the fourth and fifth basis functions being described by the same rows are generated in the same way as the third (nfsh=2) one, except with different polynomials.

The column labeled z lists the exponents for the primitive Gaussians, while the "coef" column lists the coefficient of their contribution to the linear combination comprising the basis function. Note that the uncontracted basis functions, those with jcont values of 1, have "coef" values of exactly 1. Finally, the product of the "coef" value and the normalization constant for the primitive Gaussian, N, is listed in the column labeled "rcoef."

```
 Gaussian Functions - Shell information
```

| atom | shell | jcont | ish | l | nfsh | z | coef | rcoef |
|---|---|---|---|---|---|---|---|---|
| O | 1 | 6 | 0 | 1 | 0 | 5484.6716600 | 0.0018311 | 0.8317237 |
| O | 2 | -1 | 0 | 1 | 0 | 825.2349460 | 0.0139502 | 1.5308156 |
| O | 3 | -1 | 0 | 1 | 0 | 188.0469580 | 0.0684451 | 2.4771485 |
| O | 4 | -1 | 0 | 1 | 0 | 52.9645000 | 0.2327143 | 3.2562811 |
| O | 5 | -1 | 0 | 1 | 0 | 16.8975704 | 0.4701929 | 2.7928934 |
| O | 6 | -1 | 0 | 1 | 0 | 5.7996353 | 0.3585209 | 0.9549377 |
| O | 7 | 3 | 3 | 1 | 1 | 15.5396162 | -0.1107775 | -0.6179340 |
| O | 8 | -7 | 3 | 1 | 1 | 3.5999336 | -0.1480263 | -0.2757209 |
| O | 9 | -7 | 3 | 1 | 1 | 1.0137618 | 1.1307670 | 0.8142076 |
| O | 10 | 3 | -1 | 2 | 2 | 15.5396162 | 0.0708743 | 3.1169443 |
| O | 11 | -10 | -1 | 2 | 2 | 3.5999336 | 0.3397528 | 2.4014375 |
| O | 12 | -10 | -1 | 2 | 2 | 1.0137618 | 0.7271586 | 1.0543604 |
| O | 13 | 1 | 1 | 1 | 5 | 0.2700058 | 1.0000000 | 0.2669562 |
| O | 14 | 1 | -1 | 2 | 6 | 0.2700058 | 1.0000000 | 0.2774320 |
| O | 15 | 1 | 0 | 3 | 9 | 0.8000000 | 1.0000000 | 1.1138249 |
| H1 | 1 | 3 | 0 | 1 | 15 | 18.7311370 | 0.0334946 | 0.2149354 |
| H1 | 2 | -1 | 0 | 1 | 15 | 2.8253944 | 0.2347270 | 0.3645712 |
| H1 | 3 | -1 | 0 | 1 | 15 | 0.6401217 | 0.8137573 | 0.4150514 |
| H1 | 4 | 1 | 0 | 1 | 16 | 0.1612778 | 1.0000000 | 0.1813806 |
| H1 | 5 | 1 | 0 | 2 | 17 | 1.1000000 | 1.0000000 | 1.6057611 |
| H2 | 1 | 3 | 0 | 1 | 20 | 18.7311370 | 0.0334946 | 0.2149354 |
| H2 | 2 | -1 | 0 | 1 | 20 | 2.8253944 | 0.2347270 | 0.3645712 |
| H2 | 3 | -1 | 0 | 1 | 20 | 0.6401217 | 0.8137573 | 0.4150514 |
| H2 | 4 | 1 | 0 | 1 | 21 | 0.1612778 | 1.0000000 | 0.1813806 |
| H2 | 5 | 1 | 0 | 2 | 22 | 1.1000000 | 1.0000000 | 1.6057611 |

The second table, an example of which follows below, shows much the same information in a slightly different format. In this table, the different functions corresponding to an l value are listed explicitly—for instance, the entries X, Y, and Z for the seventh shell correspond

to $p_x$, $p_y$, and $p_z$ orbitals. The only new information concerns the factors "rmfac," which may be needed to calculate each primitive Gaussian's contribution to the basis function. Sometimes, for l = 2 and higher, the value of "rcoef" calculated for the first primitive in the shell is different than it would have been if it had been calculated for a different primitive in that shell, and the "rmfac" values provide a way around that problem.

```
 Gaussian Functions - Normalized coefficients
```

| atom | shell | type | fn | z | rcoef | rmfac | rcoef*rmfac |
|---|---|---|---|---|---|---|---|
| O | 1 | S | 1 | 5484.671660 | 0.831724 | 1.000000 | 0.831724 |
| O | 2 | S | 1 | 825.234946 | 1.530816 | 1.000000 | 1.530816 |
| O | 3 | S | 1 | 188.046958 | 2.477149 | 1.000000 | 2.477149 |
| O | 4 | S | 1 | 52.964500 | 3.256281 | 1.000000 | 3.256281 |
| O | 5 | S | 1 | 16.897570 | 2.792893 | 1.000000 | 2.792893 |
| O | 6 | S | 1 | 5.799635 | 0.954938 | 1.000000 | 0.954938 |
| O | 7 | S | 2 | 15.539616 | -0.617934 | 1.000000 | -0.617934 |
| O | 8 | S | 2 | 3.599934 | -0.275721 | 1.000000 | -0.275721 |
| O | 9 | S | 2 | 1.013762 | 0.814208 | 1.000000 | 0.814208 |
| O | 10 | X | 3 | 15.539616 | 3.116944 | 1.000000 | 3.116944 |
| | | Y | 4 | | | 1.000000 | 3.116944 |
| | | Z | 5 | | | 1.000000 | 3.116944 |
| O | 11 | X | 3 | 3.599934 | 2.401438 | 1.000000 | 2.401438 |
| | | Y | 4 | | | 1.000000 | 2.401438 |
| | | Z | 5 | | | 1.000000 | 2.401438 |
| O | 12 | X | 3 | 1.013762 | 1.054360 | 1.000000 | 1.054360 |
| | | Y | 4 | | | 1.000000 | 1.054360 |
| | | Z | 5 | | | 1.000000 | 1.054360 |
| O | 13 | S | 6 | 0.270006 | 0.266956 | 1.000000 | 0.266956 |
| O | 14 | X | 7 | 0.270006 | 0.277432 | 1.000000 | 0.277432 |
| | | Y | 8 | | | 1.000000 | 0.277432 |
| | | Z | 9 | | | 1.000000 | 0.277432 |
| O | 15 | XX | 10 | 0.800000 | 1.113825 | 1.000000 | 1.113825 |
| | | YY | 11 | | | 1.000000 | 1.113825 |
| | | ZZ | 12 | | | 1.000000 | 1.113825 |
| | | XY | 13 | | | 1.732051 | 1.929201 |
| | | XZ | 14 | | | 1.732051 | 1.929201 |
| | | YZ | 15 | | | 1.732051 | 1.929201 |
| H1 | 1 | S | 16 | 18.731137 | 0.214935 | 1.000000 | 0.214935 |
| H1 | 2 | S | 16 | 2.825394 | 0.364571 | 1.000000 | 0.364571 |
| H1 | 3 | S | 16 | 0.640122 | 0.415051 | 1.000000 | 0.415051 |
| H1 | 4 | S | 17 | 0.161278 | 0.181381 | 1.000000 | 0.181381 |
| H1 | 5 | X | 18 | 1.100000 | 1.605761 | 1.000000 | 1.605761 |
| | | Y | 19 | | | 1.000000 | 1.605761 |
| | | Z | 20 | | | 1.000000 | 1.605761 |
| H2 | 1 | S | 21 | 18.731137 | 0.214935 | 1.000000 | 0.214935 |
| H2 | 2 | S | 21 | 2.825394 | 0.364571 | 1.000000 | 0.364571 |
| H2 | 3 | S | 21 | 0.640122 | 0.415051 | 1.000000 | 0.415051 |
| H2 | 4 | S | 22 | 0.161278 | 0.181381 | 1.000000 | 0.181381 |

```
H2     5  X  23       1.100000      1.605761      1.000000       1.605761
          Y  24                                  1.000000       1.605761
          Z  25                                  1.000000       1.605761
```

The table is followed by a list indicating the number of electrons in each atom which are treated with an effective core potential.

### *Gaussian function list (derivatives)*

By turning this option on, you can choose to print out information about the derivatives of the basis set functions in terms of primitive Gaussians.[12] The format and information is the same as that discussed for the Gaussian function list (basis set) option immediately above.

## 5.5. FILES OUTPUT OPTIONS

This section describes the menu options from the Files window, whose button appears under the Output heading. These output options do not alter the output file, but instead generate additional files. For each of the options described below, the relevant file will appear in the same directory as the output file. Each file's name is in the form *jobname.suffix*, where the different suffixes related to each kind of file are described below.

If you make a setting from the Files window, the output from the program pre will note the non-default options chosen. This output will appear above the molecular geometry output from the same program, and will indicate the non-default values of the keywords referred to in footnotes throughout this section.

### *Gaussian-92 input deck (.g92)*

When this option is selected, a file in the format of a GAUSSIAN input file is created, with the suffix .g92.[13] The file information includes the molecular geometry, the basis set name, and the type of calculation to be performed, as well as the molecular charge and the spin multiplicity of the molecule and any relevant effective core potential information. If symmetry is turned off, that setting will be entered into the .g92 file.

12.Keyword **ip8** = 2 in **gen** section of input file.
13.Keyword **ip160** = 2 in **gen** section of input file.

For GVB calculations, you should specify GVB pairs; Jaguar will also generate a GVB initial guess, which will be included in the .g92 file. For more information on setting up GAUSSIAN input files, see section 6.6.

## *GAMESS input file (.gamess)*

To write out an input file for the program GAMESS, you can select this option.[14] The resultant file's suffix will be .gamess. The file will include the molecular geometry, the basis set, and some information on the type of calculation to be performed, as well as the molecular charge and the spin multiplicity of the molecule and any relevant effective core potential information.

## *SPARTAN archive file (.arc)*

You can use this option to generate a SPARTAN 4.0 archive file with the suffix .arc.[15]

## *Gaussian-92 basis set (.gbs)*

If this option is turned on, a .gbs file will be generated containing the basis set in a form that can be used by GAUSSIAN.[16]

## *XYZ file (.xyz)*

If you set this option, Jaguar will create a file in XYZ format with the suffix .xyz.[17] The file will contain all geometries generated during the course of the job (except that for solvated geometry optimizations, the file currently will only contain the solvated structures).

## *Molden orbitals file (.molf)*

You can use this option to produce a file with the final orbitals in a format suitable for the program Molden [90].[18]

14.Keyword **ip168** = 2 in **gen** section of input file.
15.Keyword **ip165** = 3 in **gen** section of input file.
16.Keyword **ip163** = 2 in **gen** section of input file.
17.Keyword **ip175** = 2 in **gen** section of input file.
18.Keyword **ip90** = 2 in **gen** section of input file.

# 5.6. OUTPUT OPTIONS PER ITERATION

Some output can be printed out every SCF iteration by choosing options from the Per Iteration window, whose button appears under the Output heading. The output described in this section will appear in the output file. For each SCF iteration where the described output appears, that output is listed before the usual energy data for that iteration.

Any non-default settings from the Per Iteration window will cause the output from the program pre to note the non-default options chosen. This output will appear above the molecular geometry output from the same program, and will indicate the keywords referred to in footnotes throughout this section.

## *energy components*

When this output option is off, the individual components contributing to the total energy are only printed for the final, converged result of the SCF iterations. When the option is turned on, the output includes each iteration's energy components: namely, the nuclear repulsion term, the total one-electron terms, the total two-electron terms, the electronic energy, and the total energy.[19] The orbital energies for the occupied orbitals are also provided for each iteration.

The Coulomb and exchange contributions to the total two-electron energy will be printed as well if the J and K matrices are kept separate for the calculation, as for GVB calculations and when the Core J and K option in the Methods window is turned on. In addition, for most calculations involving open shells or higher-level methods, the individual contributions from each Hamiltonian are printed for the Coulomb and exchange terms.

If the calculation involves solvation, the nuclear-electronic and kinetic terms making up the one-electron terms will also be listed, as well as the term -V/T (where V is the potential energy and T the kinetic energy) and the various contributions to the solvation energy.

## *density matrix*

If you select this option, the density matrix in atomic orbital space will be printed out for each iteration.[20] For iterations in which Fock

19.Keyword **ip17** = 2 in **gen** section of input file.
20.Keyword **ip110** = 2 in **gen** section of input file.

matrix updating is performed, using a matrix of elements calculated by taking the change in the density matrix from one iteration to the next, it is actually this differential density matrix which will be printed. The output from the program scf indicates whether Fock matrix updating was performed or not in any particular iteration.

## *All J and K matrices, AO space*

The Coulomb and exchange matrices in atomic orbital space can be printed out for each iteration by selecting this option.[21] However, by default the calculation will be performed by combining these matrices in the form 2J – K, and they may not be properly separated here if this is the case. In order to print out the true J and K matrices, you must insure that the Core J and K option in the Methods window, whose button is found in the main window, specifies that the matrices be kept separate. For GVB, DFT, LMP2, and GVB-LMP2 calculations, the J and K matrices are kept separate by default.

Since J and K are symmetric matrices, the elements from their top triangular halves are not printed.

## *Fock matrix in AO (HF) or MO (GVB) space*

The Fock matrix in atomic orbital space (for HF or DFT calculations) or molecular orbital space (for GVB calculations) can be printed by turning this option on.[22] This information is only printed for iterations where the Fock matrix is not updated. Because the Fock matrix is symmetric, the elements from its top triangular half are not printed.

## *Fock matrix in CO space*

The Fock matrix in canonical orbital space can be printed by turning this option on.[23] Because the Fock matrix is symmetric, the elements from its top triangular half are not printed.

## *GVB data: f, a, b, ci coefficients, etc.*

You may print out GVB data for the initial guess and the GVB initial guess by selecting this option.[24]

21.Keyword **ip121** = 2 in **gen** section of input file.
22.Keyword **ip122** = 2 in **gen** section of input file.
23.Keyword **ip123** = 2 in **gen** section of input file.
24.Keyword **ip149** = 2 in **gen** section of input file.

# 5.7. OUTPUT OPTIONS FOR ORBITALS

Orbital information can be printed to the output file as well. Several possible choices are available in the Orbitals window, whose button is found under the Output heading, for what, when, and how orbitals should be printed.

If you choose to print out any orbital information, the output from the program pre will note the non-default options chosen. This output will appear above the molecular geometry output from the same program, and will indicate the keywords referred to in footnotes throughout this section.

## *When:*

*after HF initial guess*

To print out orbitals used for the HF initial guess, select this option.[25]

*after GVB initial guess*

This option allows you to print out orbitals used for the GVB initial guess.[26]

*each iteration (in CO space)*

The orbitals can be printed out after each SCF iteration in canonical orbital space.[27] (Canonical orbital eigenvectors with very small eigenvalues are removed from the calculation before the SCF process.) The number of orbitals printed may depend upon the basis set, since some basis sets use five d functions for the canonical orbitals while others use six, as described in section 3.8.

*each iteration (in AO space)*

The orbitals can be printed out after each SCF iteration in atomic orbital space.[28]

25.Keyword **ip105** in **gen** section of input file.
26.Keyword **ip106** in **gen** section of input file.
27.Keyword **ip101** in **gen** section of input file.
28.Keyword **ip103** in **gen** section of input file.

*after SCF*

If this option is selected, orbitals can be printed in atomic orbital space after the SCF iterations.[29]

*after final localization*

If Boys or Pipek-Mezey localization of the wavefunction has been requested using the button found in the main interface window, you can print the orbitals after the localization procedure by selecting this option.[30]

*at end of job*

To print the orbitals at the end of the job, select this option.[31]

## What:

By default, no orbitals are printed in the output file, so the selection none will appear in the option menu labeled What.[32] If you select occupied orbitals, all occupied orbitals, including GVB natural orbitals, will be printed.[33] If the all orbitals option is selected, all occupied orbitals and ten virtual orbitals will be printed.[34] (To change the default of ten virtual orbitals, see the information on the keyword **ipvirt** in section 8.6, under *Orbital Output Keywords*. The virtual orbitals are obtained by diagonalizing $H_0 + \sum f(2J - K)$, where $f$ is the occupation of each orbital (1 for a closed shell).) Selection of GVB orbitals (nonorthog.) prints only the GVB non-orthogonal orbitals.[35]

## How:

The choices available for how to print the selected orbitals are large elements as f5.2, labels, in list,[36] all elements as f10.5, labels, in

29.Keyword **ip104** in **gen** section of input file.
30.Keyword **ip107** in **gen** section of input file.
31.Keyword **ip102** in **gen** section of input file.
32.This setting corresponds to having all of the orbital output keywords set to 1.
33.Relevant orbital output keyword set to 2, 3, 4, 5, or 6 in **gen** section of input file, depending on the format setting chosen.
34.Relevant orbital output keyword set to 7, 8, 9, 10, or 11 in **gen** section of input file, depending on the format setting chosen.
35.Relevant orbital output keyword set to 12, 13, 14, 15, or 16 in **gen** section of input file, depending on the format setting chosen.
36.Relevant orbital output keyword set to 2, 7, or 12 in **gen** section of input file, depending on which orbitals are printed.

table,[37] all elements as f19.15, in list,[38] all elements as f8.5, in list,[39] and all elements as e15.6, in table.[40] Examples of each of these style options appear below.

In the first option listed, the phrase "large elements" indicates that only coefficients larger than a particular value (generally .05) are listed. The notations "f5.2" and the like refer to standard FORTRAN formats. The word "labels" indicates that the atom identifiers (for instance, 'h2') and the basis function types (for instance, S for s, Z for $p_z$, or XX for $d_{xx}$) are shown.

The output for each style is shown in either table form or list form. When the orbital output is in table form, each function's coefficient for each orbital is shown, with the functions shown in numbered rows and the orbitals in numbered columns. When it is in list form, each orbital is listed in turn, with the basis function coefficients listed in order. For the third and fourth options, those with f19.15 and f8.5 formatting, all coefficients are listed, in order but without numbering. The three styles presented in list form also include information on the occupation and energy of each orbital.

Because GVB orbitals are not computed until some time after the Hartree-Fock initial guess, you cannot choose to print GVB non-orthogonal orbitals if you have selected after HF initial guess above. Also, note that in canonical orbital space, the labels indicating atom identifiers and basis function types are meaningless.

If you generate output for occupied orbitals or all orbitals in the f19.15 or f8.5 formats, you can use it for input in the **guess** section of an input file, which is described in greater detail in section 8.11, or for input to GAUSSIAN (guess=cards) or GVB2P5.

Here are some examples of output for each of these style options. The output shown is from output files generated from a calculation of water with a 6-31G** basis set, where the option requested under When was after SCF iterations and the option requested under What was occupied orbitals. Only the first two occupied orbitals are shown in each case, and not all functions are shown; these gaps are indicated by [...].

37. Relevant orbital output keyword set to 3, 8, or 13 in **gen** section of input file, depending on which orbitals are printed.
38. Relevant orbital output keyword set to 4, 9, or 14 in **gen** section of input file, depending on which orbitals are printed.
39. Relevant orbital output keyword set to 5, 10, or 15 in **gen** section of input file, depending on which orbitals are printed.
40. Relevant orbital output keyword set to 6, 11, or 16 in **gen** section of input file, depending on which orbitals are printed.

For the How option large elements as f5.2, labels, in list:

```
    1 Orbital Energy   -20.555133 Occupation  1.000000 Symmetry A1
       S
O    0.99
    2 Orbital Energy    -1.345597 Occupation  1.000000 Symmetry A1
       S    S    Z    S
O   -0.21 0.47 0.09 0.42
       S
H1   0.15
       S
H2   0.15
    3 Orbital Energy    -0.713206 Occupation  1.000000 Symmetry B2
[...]
```

For the How option all elements as f10.5, labels, in table:

```
                        1         2         3
 eigenvalues-        -20.55513  -1.34560   [...]
    1 O       S        0.99466  -0.21055
    2 O       S        0.02122   0.47102
   [...]
    5 O       Z        0.00155   0.08586
    6 O       S        0.00430   0.41777
   [...]
   16 H1      S        0.00000   0.14851
   [...]
   21 H2      S        0.00000   0.14851
   [...]
   25 H2      Z        0.00025  -0.01342
```

For the How option all elements as f19.15, in list:

```
   1 Orbital Energy   -20.555133 Occupation  1.000000 Symmetry A1
 0.994661070265476  0.021223773328496  0.000000000000000  0.000000000000000
 0.001550431863529  0.004301782758377  0.000000000000000  0.000000000000000
-0.000190485390547 -0.003952404680376 -0.003763985866478 -0.003807504316264
 0.000000000000000  0.000000000000000  0.000000000000000 -0.000004988565650
-0.000343482092802  0.000000000000000  0.000372571507087  0.000252040203901
-0.000004988565650 -0.000343482092802  0.000000000000000 -0.000372571507087
 0.000252040203901
   2 Orbital Energy    -1.345597 Occupation  1.000000 Symmetry A1
-0.210549363265932  0.471018758398392  0.000000000000000  0.000000000000000
 0.085862488931510  0.417774726334513  0.000000000000000  0.000000000000000
 0.031498167188452  0.001405346737926  0.006172871870042  0.008194082815896
 0.000000000000000  0.000000000000000  0.000000000000000  0.148513692384474
 0.013067257872503  0.000000000000000 -0.022047889711935 -0.013419565122871
 0.148513692384474  0.013067257872503  0.000000000000000  0.022047889711935
-0.013419565122871
   3 Orbital Energy    -0.713206 Occupation  1.000000 Symmetry B2
 [...]
```

For the How option all elements as f8.5, in list:

```
   1 Orbital Energy   -20.555133 Occupation  1.000000 Symmetry A1
 0.99466 0.02122 0.00000 0.00000 0.00155 0.00430 0.00000 0.00000-0.00019
-0.00395-0.00376-0.00381 0.00000 0.00000 0.00000 0.00000-0.00034 0.00000
 0.00037 0.00025 0.00000-0.00034 0.00000-0.00037 0.00025
   2 Orbital Energy    -1.345597 Occupation  1.000000 Symmetry A1
-0.21055 0.47102 0.00000 0.00000 0.08586 0.41777 0.00000 0.00000 0.03150
 0.00141 0.00617 0.00819 0.00000 0.00000 0.00000 0.14851 0.01307 0.00000
-0.02205-0.01342 0.14851 0.01307 0.00000 0.02205-0.01342
   3 Orbital Energy    -0.713206 Occupation  1.000000 Symmetry B2
 [...]
```

For the How option all elements as e15.6, in table:

```
           1              2              3
  1    9.946611E-01  -2.105494E-01      [...]
  2    2.122377E-02   4.710188E-01
  [...]
  5    1.550432E-03   8.586249E-02
  6    4.301783E-03   4.177747E-01
  [...]
 16   -4.988566E-06   1.485137E-01
  [...]
 21   -4.988566E-06   1.485137E-01
  [...]
 25    2.520402E-04  -1.341957E-02
```

# 5.8. THE LOG FILE

The log file, an output file which appears in the local job directory, provides information on the progress of a run. You can display the current contents of a job's log file when you click the Check Job button found in the file viewer window, which can be opened by clicking Check from the main interface window. The log file notes when each program within Jaguar is complete, as well as noting data from each SCF iteration as it is calculated. The data from the SCF iterations is shown in table form. Some of the text for the column headings should be read down rather than across.

For the table of SCF iteration information, the number of the iteration is provided first in each row, followed by a "Y" or "N" indicating whether the Fock matrix was updated or not. When the Fock matrix is updated, the changes are made using a differential density matrix whose elements simply reflect the changes in the density matrix elements from the previous iteration to the current one.

The next entry indicates whether the DIIS convergence scheme was used for that iteration, also with a "Y" or "N". The DIIS method produces a new estimate of the Fock matrix as a linear combination of previous Fock matrices, including the one calculated during that iteration. DIIS, which is enabled by default, usually starts on the second iteration, and is not used on the final iteration. If the entry in this column reads "A", it indicates that DIIS was not used for that iteration, but the density matrix was averaged.

The cutoff set for each iteration is indicated under the "icut" heading. Cutoff sets are explained in the .cutoff file description in section 9.5.

The grid column lists the grid used for that iteration, which must be one of the grid types coarse (signified by a C), medium (M), fine (F), or ultrafine (U). See the subsection *Grid and Dealiasing Function Keywords* in section 8.6, and the description of the grid file in section 9.4, for more information on grids and grid types.

The total energy for the molecule in Hartrees appears in the next column, followed by the energy change, which is the difference in energy from the previous iteration to the current one.

The RMS density change column provides the root mean square of the change in density matrix elements from the previous iteration to the current one.

Finally, the maximum DIIS error column provides a measure of convergence by listing the maximum element of the DIIS error vector. For HF calculations, the DIIS error vector is given by **FDS – SDF** in atomic orbital space, where **F**, **D**, and **S** are the Fock, density, and overlap matrices, respectively. For open shell and GVB cases, the definition of the error vector is given in reference [11].

If you are not running a default, single-point, Hartree-Fock calculation, the log file will generally contain information generated from other Jaguar programs used for the run as well. This information is often a summary of what is written to the Jaguar output file. For a more detailed description of the information in the log file, you may wish to consult the previous sections of this chapter.

After all the individual programs necessary for that job have finished running, a note will appear in the log file listing the name and location of the output file. When the job is completed, this too is noted in the log file. At this point, if you wish, you can hit the View File button at the bottom of the file viewer window, selecting the appropriate output file name and hitting OK to see the output file for the job in the interface window.

# 6. Tips and Suggestions

Once you are familiar with the basics of running Jaguar, you may wish to use certain tricks to make it more pleasant or powerful to use. This chapter's contents include information on customizing the interface, restarting jobs, and using Jaguar's interface to help set up GAUSSIAN jobs, as well as some extra suggestions for GVB calculations, geometry optimization, ESP fitting, and jobs involving transition metals.

## 6.1. CUSTOMIZING INTERFACE SETTINGS: THE jaguar.hosts FILE

Within the Jaguar home directory, there should be a file called jaguar.hosts. Jaguar uses the jaguar.hosts file to find out which other machines in your network can also run Jaguar. The file lists where the Jaguar home directories are on those machines, and what directory should be used for temporary files and directories that are created during a Jaguar job (and removed when the job is complete). The settings in the jaguar.hosts file used when you open the interface help determine the available options in the Run window. If you want to, you can make your own, personalized jaguar.hosts file (or files). *You shouldn't need to update any jaguar.hosts files when you later install other versions of Jaguar.*

The jaguar.hosts file used by the interface will be the jaguar.hosts file in the directory where you started the interface, if it exists; otherwise, it will be the jaguar.hosts file in your home directory, if that exists. If neither of these files exist, the configuration will be determined by the

default jaguar.hosts file for the system. You can always figure out which jaguar.hosts configuration file the interface is using by clicking on the About button in the main interface window, then clicking on the Schrödinger button in the About window; the jaguar.hosts configuration file currently used by the interface will be listed near the bottom of the window.

If you would like to change entries in the jaguar.hosts file, it is usually best to make and edit your own jaguar.hosts file. If no jaguar.hosts file exists in your home directory or the directory from which you start the interface, you should identify the jaguar.hosts configuration file currently used by the interface (as described above), copy this jaguar.hosts file to your home directory or the directory where you are starting the interface, and edit it there.

The following short example of a jaguar.hosts file is used to discuss the format of the lines:

```
! This is a comment in a jaguar.hosts file
host:            anny (SGI Indigo)
home:            /scr/jaguar
*temproot:       /scr
temproot:        /scr2
!
*host:           withi (DEC Alpha)
home:            /exec/jaguar
temproot:        /temp
nprocs:          8
```

For each machine listed in the jaguar.hosts file, these three items should be listed in this order:

```
host:      machine optional-comment
home:      homedir
temproot:  tempdir
nprocs:    number of processors
```

Here, *machine* is the name of a machine that can be used to run a Jaguar calculation, *optional-comment* is an optional comment that will show up in the Run window of the interface, *homedir* is the Jaguar home directory for that machine, and *tempdir* is a directory like /scr or /temp. For stand-alone workstations with multiple processors, set *nprocs* to the number of processors in the computer. For computer clusters, each node in the cluster should have an *nprocs* setting equal to the *total* number of processors in the entire cluster.

These entries in the jaguar.hosts file may be formatted with any combination of spaces and tabs, but the entire entry must be on one

line. Any number of comments may also be included in the jaguar.hosts file; comments should start with an exclamation point (!).

A '*' at the beginning of a line designates the setting in that line as the default, for cases where you might want to specify more than one possible value. For instance, in the sample file, withi is the default host, and /exec/jaguar contains the Jaguar installation for this host. The other, non-default items will be accessible from the Run window, but if you do not specifically select them there, the Jaguar run will use the default hosts and directories.

The host name does not need to include the full Internet address (e.g. withi.schrodinger.com) unless the interface host is not on the same local network. If you have installed Jaguar on multiple machines, you may need to hand-edit each machine's jaguar.hosts file to add entries for the other machines.

The directory */tempdir/user/jobname* is used during a job to store temporary files, where *user* is the user's account name on that machine and *jobname* is the name of the Jaguar job. For example, if the user erwin ran a job named "h2o" on the machine withi using the jaguar.hosts file above, the temporary directory used for the job would be /temp/erwin/h2o. (Jaguar will, by default, completely remove this subdirectory in the temporary space when the job is completed, after copying back all important files to the output directory.)

You can override the `temproot` setting in the jaguar.hosts file by using the JAGUAR_TEMP environment variable. For example, if the scratch directory designated by `temproot` becomes full with files that you don't have permission to delete, you can set JAGUAR_TEMP to a different directory and continue to run Jaguar jobs. Unlike the `temproot` setting, the setting for JAGUAR_TEMP should probably include your username, because the username will not automatically be appended. Thus, if /scratch is full and you want to use /scratch2, then create a directory for yourself on /scratch2 and set JAGUAR_TEMP to /scratch2/<your username>.

If you have different user names on the interface and calculation hosts and these machines are on separate area networks, you may need to create a jaguar.hosts file in your home directory to avoid getting an error message indicating that your login is not correct. The jaguar.hosts file should include a host line of the form

```
host:   schrod@anny.schrodinger.com (SGI Indigo)
```

where the name of the machine in the "host:" field matches that of the output from the "uname -n" command for that machine. After you

have created the jaguar.hosts file in your directory, substitute your user name on the calculation host for "schrod" and your calculation host name for "anny.schrodinger.com".

## 6.2. CUSTOMIZING THE INTERFACE APPEARANCE

First, please note that if you use the window manager twm, you may be able to improve the appearance of the interface by adding the line

```
DecorateTransients
```

to your .twmrc file in your home directory. Without this line, the borders for the interface windows may not appear.

Some of the X resources used by the interface are determined by entries in the jaguar.style file. These X resources only affect the appearance the interface, not its functionality. If you want to change some or all of these resources for all users, copy the jaguar.style file from the Jaguar home directory to your home directory and edit the ones you want to change. Note that if a fixed-size font such as Courier is the default for a particular resource, you should probably only replace it with another fixed-size font, and that fonts which include the special ASCII characters above 127 (such as Å) are preferable.

The jaguar.style file used by the interface will be the jaguar.style file in the directory where you started the interface, if it exists; otherwise, it will be the jaguar.style file in your home directory, if that exists. If neither of these files exist, the configuration will be determined by the jaguar.style file in the Jaguar home directory.

On some machines, when you iconify the main Jaguar window, the Jaguar icon will not completely show. To improve the icon appearance, you can change the maximum size for icons by putting the appropriate line in a file called .Xdefaults in your home directory. For instance, for the mwm window manager, this line could read "Mwm*iconImageMaximum: 85x67."

## 6.3. TIPS FOR VARIOUS TYPES OF JOBS

This section contains information you may find useful for improving SCF convergence, running GVB jobs and optimizations, and fitting charges.

## *Organometallics and Other Difficult-to-Converge Systems*

Generally, Hartree-Fock wavefunctions for simple organic molecules converge in fewer than 10 iterations, while complex calculations involving higher-level methods or open shells may take a few extra iterations. Molecules which include transition metals generally converge more slowly, however. Make sure your job has really converged and did not simply end because it reached the maximum number of SCF iterations, a number set in the Methods window.

If a job gives poor SCF convergence, you can try either modifying the convergence methods used or improving the initial guess. To modify the convergence methods, try any or all of the following settings:

- Try setting "iacscf" equal to 1, 2, 3, or 4 (see table 8.6.22 for descriptions of each number's function).

- Select GVB-DIIS from the Convergence scheme option menu in the Methods window. Generally, DIIS is the better choice, but the GVB-DIIS convergence scheme sometimes leads to convergence when DIIS does not.

- Set the SCF level shift in the Methods window to 0.5 or 1.0. The higher the setting, the more the virtual orbitals' energies are increased before diagonalization, and the more the mixing of the real and virtual orbitals is reduced. High SCF level shifts sometimes slow convergence by several iterations, but can often help otherwise intractable cases to converge. Because jobs with SCF level shifts are slightly more likely to converge to excited states, you may also wish to restart these jobs without any SCF level shift.

- Change the Accuracy level setting in the Methods window to ultrafine. This setting will cause the job to use denser pseudospectral grids and tighter cutoffs, and generally increases computational costs by a factor of two to three.

- If the calculation is a DFT job, use finer DFT grids. You can make this setting from the Grid density option menu in the DFT window. This setting will also increase the computational cost.

For transition-metal-containing systems, particularly organometallics, you can often obtain superior results by improving the initial guess wavefunction. Jaguar automatically generates high-quality initial guesses for transition-metal-containing compounds; if you supply the

program with information about the charges and spins of the "fragments" in the compounds, it will use that information when generating the guess. Here, a fragment is defined as either a collection of one or more transition metals that are bonded together, or one or more non-transition-metal atoms bonded together. Put another way, each fragment is simply a group of atoms that would be bonded together even if all bonds between transition metal atoms and non-transition-metal atoms were broken. Typically, the system is broken into ligand fragments and transition metal fragments, or adsorbate fragments and cluster fragments. For example, for ferrocene, the iron atom is one fragment, and the two cyclopentadienyl ligands are two additional fragments.

To supply Jaguar with information on charges and spins for its high-quality initial guess for a transition-metal-containing system, you need to edit the input file, either from the Edit Job window in the Jaguar interface (which is accessible by clicking on the Edit Input button) or from a terminal window. First, add the line

```
iguess=25
```

between the "&gen" and the next "&" sign. Next, add these lines to the bottom of the input file:

```
&atomic
atom   formal   multip
&
```

(The exact number of spaces between words does not matter.)

Finally, fill in information for each fragment under the headings "atom," "formal," and "multip." You should add a single line for each fragment with a formal charge and/or non-singlet spin multiplicity. The first entry in the line (under the heading "atom") should be the atom label of *any* atom in the fragment. The next entry (under the heading "formal," and separated from the first entry by one or more spaces) should be the formal charge of the entire fragment. The third entry (under the heading "multip") should be the spin multiplicity of the fragment. If C1 is in one ring of ferrocene and C2 is in the other ring, then the following &atomic section could be used to help generate the initial guess:

```
atom   formal   multip
Fe      +2       1
C1      -1       1
C2      -1       1
&
```

Fragments with no formal charge and singlet spin (water, for example) do not need to be listed in the **atomic** section, because Jaguar will assume a default formal charge of 0 and multiplicity of 1 for each fragment. Note, however, that any charge or spin multiplicity settings in the **atomic** section must be compatible with any settings for overall charge and spin specified by the "molchg=" and "multip=" keywords in the **gen** section. For more information about the **atomic** section, please see section 8.9.

After saving the input file with the **iguess** setting and **atomic** section, you can run it in Jaguar in the usual manner.

## *GVB Calculations: GVB Pair Selection*

For most molecules, Lewis dot structures give a reasonable idea of what GVB pairs you should consider setting. If you want to automatically assign pairs by Lewis dot structure for input files generated and submitted outside the interface, see the *GVB and Lewis Dot Structure Keywords* subsection of section 8.6. You do not have to assign all possible GVB pairs. You can set GVB pairs in any order.

If you are studying a dissociating bond, you should assign all reasonable GVB pairs for that bond. For some purposes, such as for dipole moment calculations, you may find that assigning only pairs for bonds between two different atoms will suffice. Bonds to hydrogen atoms can also be ignored for some cases.

You should not assign GVB lone pairs if you are using a minimal basis set, since the basis set will not have enough degrees of freedom to handle the lone pair. When assigning lone pairs, you should only put one GVB lone pair on atoms from the nitrogen group, two for those from the oxygen group, three for the fluorine group, and one for the carbon group. In the last case, assigning lone pairs is only reasonable when the atom is bonded to only two neighbors. If you assign one GVB lone pair for an atom, you should also assign any other possible GVB lone pairs on that atom.

## *Geometry Optimization*

If you are performing a geometry optimization and are not starting from a high-quality initial molecular structure, you may wish to do a "quick and dirty" calculation to obtain a somewhat better geometry, then perform a more accurate calculation by starting with the results you have generated already. For example, if you wanted to perform an LMP2 geometry optimization, you could start by performing a Hartree-Fock geometry optimization, then restart the calculation by

using the HF results in an LMP2 geometry optimization. See Section 6.4 for a description of restarting calculations and incorporating previous results in a later run.

Whenever you are doing a geometry optimization, make sure that you really do obtain a converged structure; the run will end before converging if you reach the maximum number of iterations allowed (as set in the Optimization window). If it did not reach convergence, you can restart the run, as described in Section 6.4.

### *Electrostatic Potential Charge Fitting*

It is probably best not to constrain electrostatic potential charge fitting to reproduce multipole moments higher than the dipole moment, because the errors in fitting the Coulomb field outside the molecule are likely to be high. Fitting to the dipole moment is usually safe; in fact, even without this constraint, the dipole moment resulting from the fitted charges is generally similar to that calculated from the wavefunction.

## 6.4. RESTARTING JOBS AND USING PREVIOUS RESULTS

Sometimes, you may find it useful to restart a job, either because you want to refine the results and do not want to start from the beginning of the calculation, because you want to alter the calculation slightly but want to use an initial guess or geometry from the previous run, or because you encountered some sort of problem that prevented the job from finishing. New input files, which are also called restart files, generated during each job can be used to restart the jobs. These files are automatically written to your local job directory at the end of a run; if the run did not complete, you can usually find the new input file by following the directions under *Finding the Restart File in the Temp Directory* below.

A new input file, or restart file, appears in the local job directory when any Jaguar job is completed. This file contains all the information needed for a new run incorporating the results from the first run. This file will contain the same job settings you made for the original input file for the job, but will also contain the results of the job—the final wavefunction, the final geometry, and the like. Thus, if you want to restart the calculation with the wavefunction and other data already calculated, you can just read in the new input file. The file's name is

<job>.**.in, where the asterisks represent a two-digit number. This number is 01 if the name of the input file for the job during which it was generated is not in the form <job>.**.in, and is otherwise set to the number after that assigned to the current input file. These files will overwrite any other existing files of the same name.

As an example, if you run the job h2o, the restart file generated during the run will be called h2o.01.in. You could then read this file into the interface, as described in section 2.4, and use it to continue on with the calculation, possibly after making some changes to the calculation requested. The new input file generated during this second run would be called h2o.02.in.

If you want to start a new job where the previous job left off, you need only read the new input file in, then make any changes you think are necessary—for example, you could change the SCF energy convergence criterion from the Methods window, whose button appears in the main window. Similarly, if you wish to perform an additional calculation once a geometry has been optimized, you can read in the restart file as input for the second job and make any necessary changes to it, such as selecting a GVB calculation instead of Hartree-Fock. section 2.4 contains information on reading input files into the interface. Please see Chapter 8 if you would like more information on input files.

Please note that if you restart a run, you may not get exactly the same results as you would if you had simply performed a longer run in the first place, even if the calculation type is the same. The methods used in Jaguar sometimes use data from previous iterations, if this information is available, but the data may not be stored in the new input file. For example, the DIIS convergence scheme uses Fock matrices from all previous iterations for the run, and Fock matrices are not stored in new input files. However, calculations should ultimately converge to the same answer within a standard margin of error whether they are restarted or not.

## *Finding the Restart File in the Temp Directory*

If your run aborted or was killed before completion, and you want to restart the calculation or start another calculation where that one left off, you can look for a file called "restart.in" in the temporary space used for the job. The file will be located in a subdirectory whose name is the same as the job's, and which is found within the temp directory for the job, which was listed in the Run window.

By default, the restart.in file is written out at the end of the Jaguar programs for calculating the initial guess, performing the SCF iterations, and calculating a new geometry for geometry optimizations, as well as at the end of each SCF iteration. (To turn off restart.in file generation, the input file output keywords **ip151** and/or **ip152** in the **gen** section would need to be set to 0.) The restart.in file overwrites itself each time, so that the final version is written either at the end of the run or just prior to any problems encountered.

# 6.5. USING JAGUAR WITH MOPAC

Jaguar includes a basic command-line interface to the program MOPAC 6.0 [91]. The interface makes it easy to run MOPAC jobs from Jaguar input and to incorporate MOPAC results (such as optimized geometries or Hessians) into Jaguar input files.

For the Jaguar/MOPAC interface, all commands are in the format

```
jaguar mopac options input
```

where *options* indicates optional flags and *input* indicates the full name of either a MOPAC input file whose name ends in the suffix ".dat" or a Jaguar input file whose name ends in the suffix ".in".

Features of the Jaguar/MOPAC interface that can be accessed with commands of this sort are described below. You can also view some help for the Jaguar/MOPAC interface by entering the command

```
jaguar mopac -help
```

## *Running MOPAC*

From the Jaguar/MOPAC interface, the command to run MOPAC using input from a Jaguar input file is

```
jaguar mopac jobname.in
```

where *jobname*.in is a Jaguar input file. The MOPAC input file *jobname*.dat is generated and then submitted to MOPAC, which produces several files. The MOPAC output for the job is written to the file *jobname*.out.

If you have a MOPAC input file named *jobname*.dat, you can submit it to MOPAC through the Jaguar/MOPAC interface with the command

```
jaguar mopac jobname.dat
```

Once again, the main output for the MOPAC job is written to the file *jobname*.out.

You can use MOPAC keywords for MOPAC runs by adding the flag "-k" to the command, followed by the desired MOPAC keyword(s). If you are specifying more than one MOPAC keyword, the keywords should be within quotation marks, in this manner:

```
jaguar mopac -k "keyword1 keyword2" input
```

where *keyword1* and *keyword2* are MOPAC keywords and *input* is either a Jaguar input (.in) file or a MOPAC input (.dat) file.

If you want to obtain a MOPAC input file to run later in MOPAC, you can do so with the command

```
jaguar mopac -n jobname.in
```

which creates a MOPAC input file *jobname*.dat from a Jaguar input file *jobname*.in.

## Adding MOPAC Geometries to Jaguar Input

You can use the Jaguar/MOPAC interface to replace the geometry in a Jaguar input file with a MOPAC-optimized one by using the "-geom" option. For instance, the command

```
jaguar mopac -geom jobname.in
```

takes the Jaguar input *jobname*.in, generates from it the MOPAC input file *jobname*.dat, runs MOPAC with the input *jobname*.dat, extracts the resultant MOPAC-optimized geometry from the MOPAC output file *jobname*.out, and uses it when creating a new Jaguar input file *jobname*_mopac.in that is identical to the original input file *jobname*.in except for the geometry.

## Using MOPAC Hessians in Jaguar Input

With the "-hess" option to the Jaguar/MOPAC interface, you can add a MOPAC Hessian to a Jaguar input file, so that it will be used as the initial Hessian for a subsequent Jaguar optimization. Before using the "-hess" option, however, you should decide whether you want the geometry in the resultant Jaguar input file to be the MOPAC-optimized geometry or the geometry in the original input file submitted to the "jaguar mopac" command.

To create a Jaguar input file with a MOPAC-optimized geometry and Hessian, enter a command in the format

```
jaguar mopac -hess -geom input
```

or, equivalently,

```
jaguar mopac -hess input
```

where *input* is either a Jaguar input (.in) file or a MOPAC input (.dat) file. A new Jaguar input file *jobname*_mopac.in will be created, containing the MOPAC-optimized geometry and Hessian; if *input* was a Jaguar input file, *jobname*_mopac.in will also contain other job settings from the file *input*.

To add a MOPAC Hessian to a Jaguar input file without actually changing the geometry, enter a command in the format

```
jaguar mopac -align -hess jobname.in
```

or, equivalently,

```
jaguar mopac -align jobname.in
```

where *jobname*.in is a Jaguar input file. A new Jaguar input file *jobname*_mopac.in will be created, containing the MOPAC-optimized Hessian and any other job settings from the file *jobname*.in. The "-align" flag tells the Jaguar/MOPAC interface to orient the MOPAC Hessian and the initial geometry the same way.

## *Specifying Different File Names*

As described above, if you run a MOPAC job from Jaguar input *jobname*.in, the MOPAC input file generated is called *jobname*.dat by default. You can specify a different name for the MOPAC input file by using the "-m" option. For instance, the command

```
jaguar mopac -m mopjob.dat jagjob.in
```

would create a MOPAC input file called mopjob.dat from the Jaguar input file jagjob.in, and run MOPAC using mopjob.dat as input.

Similarly, if you use the Jaguar/MOPAC interface to create a new Jaguar input file that incorporates a MOPAC geometry and/or Hessian, you can specify the name of the new Jaguar input file with the "-j" option. For example, the command

```
jaguar mopac -geom -j newjagjob.in jagjob.in
```

takes the Jaguar input file jagjob.in, generates from it the MOPAC input file jagjob.dat, runs MOPAC with the input jagjob.dat, extracts the resultant MOPAC-optimized geometry from the MOPAC output file jagjob.out, and uses it when creating a new Jaguar input file newjagjob.in that is identical to the original input file jagjob.in except for the geometry.

# 6.6. SUGGESTIONS FOR GAUSSIAN USERS

We recognize that some Jaguar users also use GAUSSIAN for calculations. Therefore, Jaguar can generate or read GAUSSIAN input files. If you plan to perform GVB calculations with GAUSSIAN, you will find this feature particularly useful, since you can use Jaguar to generate a high-quality GVB initial guess automatically.

## *Generating GAUSSIAN Input Files with Jaguar*

You can use Jaguar's interface as a convenient tool to create GAUSSIAN input files, if you wish. The output file produced from the Jaguar run, whose name will end in .g92, can be used as a GAUSSIAN input file. The .g92 file will request an HF or ROHF (restricted open-shell Hartree-Fock) calculation, whichever is appropriate for the number of electrons in the system, unless you choose to specify another method. Details applying only to constructing an input file for a GVB calculation are discussed below.

To create a .g92 file, turn on the Gaussian-92 input deck (.g92) option, which is found in the Files window, whose button appears under the Output heading. If you are just creating a GAUSSIAN input file and you do not want to use Jaguar to generate a converged wavefunction, you can save some time by using the Edit Job window to add the keyword setting **igonly**=1 (initial guess only) to the **gen** section of the input file.

The information in the .g92 file will depend on the information you have provided to the interface. The file will always contain a molecular geometry (in Cartesian coordinates and Angstroms); instructions for how to input geometries into the Jaguar interface are available in section 2.2. The file will also specify the molecular charge and the spin multiplicity of the molecule. If you want either of these values to be non-zero, you can make the appropriate settings near the bottom of the interface. You can also set the name of the basis set you want to provide in the .g92 file (for example, STO-3G) by using the Basis Set

window, whose button is also in the main window. (The default basis set choice is 6-31G**.)

If you want, you can turn off symmetry by using the Symmetry button found in the main window.

To actually generate the .g92 file, you need to run the Jaguar job you have just specified. See section 2.7 for information on running jobs.

*Additional Details for Making Input Files for GVB Calculations*

To set up the .g92 file for a GVB calculation, you should use the default setting, Compute from HF initial guess, from the GVB initial guess option menu, which is in the Methods window accessed by the Methods button in the main window. You should specify the GVB pairs in the GVB window, as well. See section 3.3 for information on setting up GVB calculations.

If you have selected a GVB calculation, the interface will automatically turn off the use of symmetry, and the .g92 file will also specify 'nosymm.' You may wish to delete this setting from the .g92 file after it is produced.

The .g92 file will also contain a Jaguar-generated initial guess if you have selected a GVB calculation, and will note that this trial wavefunction is to be used as an initial guess for the GAUSSIAN run ("guess=cards"). If you have chosen to do an initial-guess-only calculation, as described above, the initial guess will generated from Jaguar's GVB initial guess routine. Otherwise, the initial guess provided in the .g92 file will be the final wavefunction resulting from the Jaguar SCF calculation performed starting from the GVB initial guess.

*Other Non-Interface Jaguar Options for the .g92 File*

You can use a Jaguar input file to run a Jaguar job which generates a .g92 file. See Chapter 8 for a description of input files. Selecting the Gaussian-92 input deck (.g92) interface output option described above corresponds to setting the output keyword **ip160** to 2 in the **gen** section of the input file.

You can create or edit Jaguar input files by hand, making keyword settings corresponding to all of the relevant options described above; see Chapter 8 for details. If you wish, you can make some of the desired settings with the interface, use the Save window to save a Jaguar input file, and edit it by hand later to set other keywords.

You can generate additional information for the .g92 file by setting the output keyword **ip160** in the **gen** section of the input file to 3, 4, or 5. Setting this keyword to 3 lets you provide an initial guess within the .g92 file (as described for GVB calculations above) even if you are doing a non-GVB calculation. Setting it to 5 allows you to explicitly provide the basis set itself, rather than just the basis set name, within the .g92 file. This option is useful for specifying basis sets which are included in Jaguar but not in GAUSSIAN. Setting **ip160** to 4 allows you to include both the initial guess and the basis set in the .g92 file. These options will also appear in the interface in the future.

## *Getting Basis Sets or Orbitals for GAUSSIAN*

The preceding subsection describes how to generate basis sets or orbitals for a GAUSSIAN input file. You can also output a basis set in the format used by GAUSSIAN by turning on the Gaussian-92 basis set (.gbs) option, which is found in the Files window, whose button is under the Output heading. The output will appear in a file whose name will end with .gbs.

You can output orbitals from Jaguar in the format used by GAUSSIAN (for its 'guess=cards' option) by choosing to output the appropriate orbitals from the Orbitals window, which is described in section 5.7. You must choose the f19.15 or f8.5 format from the How option menu.

## *Using GAUSSIAN Files as Jaguar Input*

GAUSSIAN input files can be read in to the Jaguar interface, which will get the molecular geometry from them, and will also turn symmetry off for the calculation or turn on electrostatic potential fitting to atomic centers if the GAUSSIAN input file requests either of those options. Any other Jaguar settings will take on their default values. For information on scanning in GAUSSIAN input files as Jaguar input, see section 2.4.

# 7. Theory

This chapter contains a description of some of the theory behind the methods used in Jaguar. Section 7.1 describes the pseudospectral method itself. Section 7.2, Section 7.3, and Section 7.4 describe GVB, GVB-RCI, and LMP2 calculations and how the pseudospectral method improves computational scaling and efficiency for these methods. Section 7.5 contains a brief description of density functional theory. Chapter 3 includes information about performing Jaguar calculations using the techniques described here.

## 7.1. THE PSEUDOSPECTRAL METHOD

Like conventional ab initio electronic structure codes, Jaguar solves the Schrödinger equation iteratively, using self-consistent field methods to calculate the lowest-energy wavefunction within the space spanned by the selected basis set. For calculations on large molecules, both conventional and pseudospectral techniques must recalculate key integral terms for each SCF iteration, since storage costs for these terms are prohibitive.

However, most of the fundamental integrals calculated in the pseudospectral method [1-9] are computed in physical space, on a grid, rather than in the spectral space defined by the basis functions. The pseudospectral method takes the density matrix from the wavefunction guess used at the beginning of each SCF iteration and the values of the integrals upon the grid points and manipulates them to produce the necessary operators on the grid, then assembles the Fock matrix by transforming these components back into spectral space, where the

Fock matrix is employed in the usual way to generate the wavefunction for the next iteration.

For medium and large molecules, the additional overhead the pseudospectral method requires to compute the information needed for the transformation between physical and spectral space is vastly outweighed by the advantages of evaluating the integrals in physical space. The matrix needed for the transform from physical to basis set space [7] can be assembled before the SCF iterations by calculating the least-squares operator $\boldsymbol{Q}$, which is given by the equation

$$\boldsymbol{Q} = \boldsymbol{S}[\boldsymbol{R}^{\dagger}\boldsymbol{w}\boldsymbol{R}]^{-1}\boldsymbol{R}^{\dagger}\boldsymbol{w} \tag{1}$$

where $\boldsymbol{S}$ is the usual analytic overlap matrix, $\boldsymbol{R}$ is a matrix of fitting functions evaluated at the grid points, and $\boldsymbol{w}$ is a diagonal matrix of grid weights. The fitting functions used to construct the matrix $\boldsymbol{R}$ include both basis functions and dealiasing functions, which are chosen in order to span the function space represented by the grid more completely than the basis functions alone. The operator $\boldsymbol{Q}$ can be calculated for the relevant basis functions using several different sets of grid points, where each set of points defines a grid type, ranging from coarse to ultrafine depending on the number of grid points.

In practice, not all possible $\boldsymbol{Q}_{\boldsymbol{ig}}$ elements are calculated for each basis function $\boldsymbol{i}$ and each grid point $g$, because most basis functions drop off sharply enough that they have no significant value on some or most grid points. These functions are classified as short-range functions and are grouped together by atom, while the remaining functions are classified as long-range functions, which are all considered to be in one single group [13].

Since $\boldsymbol{Q}$ does not depend on the wavefunction itself, it can be fully computed before the SCF procedure. However, since the $\boldsymbol{Q}$ for each grid type contains $N_{basis}$ x $N_{grid}$ elements, where $N_{basis}$ is the number of basis functions and $N_{grid}$ the number of grid points (which is generally larger than $N_{basis}$), we sometimes reduce memory demands by only computing and storing the $N_{basis}$ x $N_{fit}$ matrix $\boldsymbol{S}[\boldsymbol{R}^{\dagger}\boldsymbol{w}\boldsymbol{R}]^{-1}$ in the program rwr, for cases where the $\boldsymbol{Q}$ for that grid type is only needed for one SCF iteration. We then assemble the full $\boldsymbol{Q}$ during the SCF iteration for which it is needed.

After the Jaguar program rwr has generated the $\boldsymbol{Q}$ or $\boldsymbol{S}[\boldsymbol{R}^{\dagger}\boldsymbol{w}\boldsymbol{R}]^{-1}$ matrix, the program scf takes the initial orbitals and iteratively modifies them with the pseudospectral method until convergence. This process involves calculating the values of the necessary integrals on

the grid points, and actually assembling the Fock matrix from the computed information. The three-center, one-electron pseudospectral integrals on the grid points are defined by

$$A_{klg} = \int \frac{\varphi_k(1)\varphi_l(1)}{r_{1g}} dr_{1g} \tag{2}$$

where $\varphi_k$ and $\varphi_l$ are basis functions and the index $g$ represents a grid point. These integrals are calculated for all combinations of basis functions and grid points not eliminated by cutoffs, and the Fock matrix is assembled from its Coulomb and exchange matrix components $J_{ij}$ and $K_{ij}$, which are calculated in physical space and transformed back into spectral space by the following equations:

$$J_{ij} = \sum_g Q_{ig} \left[ \sum_{kl} A_{klg} D_{kl} \right] R_{jg} \tag{3a}$$

$$K_{ij} = \sum_g Q_{ig} \left[ \sum_n A_{jng} \sum_m D_{nm} R_{mg} \right] \tag{3b}$$

where $D$ is the usual spectral space density matrix, $R_{jg}$ is the value of the function $j$ at grid point $g$, and $A_{klg}$ is given by Equation (2). The grid points used for each SCF iteration are determined by the grid type (coarse, medium, fine, or ultrafine) chosen for that iteration. The number of arithmetic operations involved in the assembly of the matrices $J$ and $K$ in Equation (3a) and Equation (3b) scales formally as $N^3$, as opposed to the $N^4$ scaling for the matrix assembly in the conventional spectral space algorithm.

Jaguar actually uses the pseudospectral method described above for the majority of the computationally intensive two-electron integral terms, but calculates the one-electron and some of the largest and most efficiently computed two-electron terms analytically [13]. More specifically, for the Coulomb matrix elements, we calculate the analytic terms

$$\sum_{kl} (ij|kl) D_{kl}$$

for cases in which $i, j, k,$ and $l$ meet certain cutoff criteria and the term $(ij|kl)$ is of the form $(aa|aa)$, $(aa|ab)$, $(aa|bb)$, $(ab|ab)$, or $(aa|bc)$, where $a$, $b$, and $c$ indicate the atom upon which the function is centered. Similar correction terms are computed for the exchange operator, as detailed in ref. [13]. The corresponding pseudospectral

terms, as dictated by Equation (3a) and Equation (3b) for the appropriate ***i, j, k,*** and ***l*** choices, must be subtracted from the pseudospectral ***J*** and ***K*** elements as well. This combined pseudospectral/analytic approach allows Jaguar to take advantage of the strengths of both methods, since it can largely maintain the pseudospectral method speedups for a particular grid, and can also use a coarser grid than a purely numerical calculation would allow.

# 7.2. PSEUDOSPECTRAL IMPLEMENTATION OF THE GVB METHOD

The pseudospectral method has also been extended to electron correlation methods, with a particular focus on Generalized Valence Bond (GVB) [20] calculations. Highly refined GVB initial guess [14] and convergence [11] algorithms have been automated within Jaguar, allowing the scaling advantages resulting from the pseudospectral method to be maintained for GVB calculations. The method yields very accurate excitation energies, rotational barriers, and bond energies for many molecules, and GVB calculations with Jaguar are typically 10 to 100 times more efficient than the best conventional GVB programs, even for molecules as small as ten atoms [6].

In the GVB approach, each bond or other electron pair is described by two non-orthogonal orbitals, whose contributions to the bond description are obtained variationally. The bond description can thus change smoothly from a description with two atomic-like orbitals at large bond distances to a description with bond-like orbitals at short distances. This improvement over Hartree-Fock, which treats bonds as having equal amounts of covalent and ionic character, allows GVB to describe charge transfer reactions and bond breaking and formation accurately, and also gives better results for other molecular properties than an HF treatment alone can provide.

The goal of a GVB calculation, then, is to obtain pairs of GVB orbitals $\psi_{pa}$ and $\psi_{pb}$, where p ranges from 1 to the number of GVB pairs $N_{gvb}$, that lead to a minimum energy for the molecular wavefunction

$$\Psi = \prod_{p=1}^{N_{gvb}} (\psi_{pa}\psi_{pb} + \psi_{pb}\psi_{pa})(\alpha\beta - \beta\alpha) \; . \qquad (4)$$

For a given p, the orbitals $\psi_{pa}$ and $\psi_{pb}$ form a pair that describes a particular bond or other pair of electrons. Under the perfect pairing restriction, the GVB orbitals within a pair are not orthogonal, although they are each orthogonal to all GVB orbitals in other pairs. For computational purposes, it is useful to form orthogonal GVB natural orbitals $\psi_{pg}$ and $\psi_{pu}$ from the GVB orbitals $\psi_{pa}$ and $\psi_{pb}$ and their overlap $S_p$, as follows:

$$\psi_{pg} = \frac{(\psi_{pa} + \psi_{pb})}{\sqrt{2(1 + S_p)}} \tag{5a}$$

$$\psi_{pu} = \frac{(\psi_{pa} - \psi_{pb})}{\sqrt{2(1 - S_p)}} \tag{5b}$$

The $\psi_{pg}$ orbitals generally have a bonding character, while the $\psi_{pu}$ orbitals are anti-bonding. The contribution to the GVB wavefunction from each pair is given by

$$(C_{pg}\psi_{pg}\psi_{pg} - C_{pu}\psi_{pu}\psi_{pu})(\alpha\beta - \beta\alpha) \tag{6}$$

where the GVB configuration interaction (CI) coefficients $C_{pg}$ and $C_{pu}$ satisfy the following equations:

$$\frac{C_{pg}}{C_{pu}} = \frac{(1 - S_p)}{(1 + S_p)} \tag{7a}$$

$$C_{pg}^2 = C_{pu}^2 \tag{7b}$$

Solving for the optimal GVB orbitals is therefore a matter of determining both the GVB natural orbitals and the GVB CI coefficients that minimize the energy of the GVB wavefunction. This energy is given by the equation

$$\mathrm{E} = \sum_{\mu}^{2N_{gvb}} 2C_{\mu}^2 h_{\mu\mu} + \sum_{\mu\nu}^{2N_{gvb}} (a_{\mu\nu}J_{\mu\nu} + b_{\mu\nu}K_{\mu\nu}) \tag{8}$$

where $\mu$ and $\nu$ range over all GVB natural orbitals (bonding and anti-bonding), and where these orbitals are expanded in terms of the basis functions, as shown here:

$$\psi_u = \sum_i^{N_{basis}} c_{i\mu}\varphi_i \tag{9}$$

The terms $h_{\mu\mu}$, $J_{\mu\nu}$, and $K_{\mu\nu}$ are defined by:

$$h_{\mu\mu} = \langle\psi_\mu|\boldsymbol{h}|\psi_\mu\rangle = \sum_{ij}^{N_{basis}} c_{i\mu}c_{j\mu}h_{ij} \tag{10a}$$
$$= \sum_{ij}^{N_{basis}} c_{i\mu}c_{j\mu}\langle i|\boldsymbol{h}|j\rangle$$

$$J_{\mu\nu} = (\mu\mu|\nu\nu) = \langle\psi_\nu|J_\mu|\psi_\nu\rangle \tag{10b}$$
$$= \sum_{ij}^{N_{basis}} c_{i\nu}c_{j\nu}J_{ij}^{\mu}$$

$$K_{\mu\nu} = (\mu\nu|\mu\nu) = \langle\psi_\nu|K_\mu|\psi_\nu\rangle = \sum_{ij}^{N_{basis}} c_{i\nu}c_{j\nu}K_{ij}^{\mu} \tag{10c}$$
$$= \sum_{ij}^{N_{basis}} c_{i\nu}c_{j\nu}\sum_{kl}^{N_{basis}} c_{k\mu}c_{l\mu}(ik|jl)$$

and the quantities $a_{\mu\nu}$ and $b_{\mu\nu}$ obey the following rules:

$$a_{\mu\mu} = C_\mu^2\,,\ b_{\mu\mu} = 0\ ; \tag{11a}$$

$$a_{\mu\nu} = 0\,,\ b_{\mu\nu} = -C_\mu C_\nu \tag{11b}$$
for $\mu$ and $\nu$ in the same pair ($\mu \neq \nu$); and

$$a_{\mu\nu} = 2C_\mu^2 C_\nu^2\,,\ b_{\mu\nu} = -C_\mu^2 C_\nu^2 \tag{11c}$$
for $\mu$ and $\nu$ in different pairs.

Examining the variation of the energy E with respect to the basis set coefficients $c$ gives the equations for the Fock operator corresponding to each GVB natural orbital:

$$F_{ij}^{\nu} = C_{\mu}^{2} h_{ij} + \sum_{\nu}^{2N_{gvb}} (a_{\mu\nu} J_{ij}^{\nu} + b_{\mu\nu} K_{ij}^{\nu}) \tag{12}$$

Each orbital's Fock operator thus depends on the other orbitals' Coulomb and exchange operators.

At the beginning of each SCF iteration, the scf program is provided with a set of proposed natural orbitals and a set of CI coefficients that dictate the contribution of each natural orbital to the GVB orbitals. For that set of GVB natural orbitals, the program first solves for revised CI coefficients by evaluating the Coulomb and exchange matrix elements for those orbitals and diagonalizing the two-by-two operators $\boldsymbol{Y}^{p}$ in the basis of the two natural orbitals in pair $p$, as described by these equations:

$$Y^{p} C^{p} = C^{p} E^{p} \tag{13a}$$

$$\begin{matrix} p \\ pg,\, pg \end{matrix} = h_{pg,\, pg} + \frac{1}{2} J_{pg,\, pg} + \tag{13b}$$

$$\sum_{q \neq p}^{N_{gvb}} C_{qg}^{2} (2J_{qg,\, pg} - K_{qg,\, pg}) + \sum_{q \neq p}^{N_{gvb}} C_{qu}^{2} (2J_{qu,\, pg} - K_{qu,\, pg})$$

$$\begin{matrix} p \\ pu,\, pu \end{matrix} = h_{pu,\, pu} + \frac{1}{2} J_{pu,\, pu} + \tag{13c}$$

$$\sum_{q \neq p}^{N_{gvb}} C_{qg}^{2} (2J_{qg,\, pu} - K_{qg,\, pu}) + \sum_{q \neq p}^{N_{gvb}} C_{qu}^{2} (2J_{qu,\, pu} - K_{qu,\, pu})$$

$$Y_{\mu\nu}^{p} = \frac{1}{2} K_{\mu\nu} \qquad (\text{for } \mu \neq \nu) \tag{13d}$$

In practice, since the CI coefficients mutually interdependent, they are determined using a self-consistent iterative procedure.

Next, holding the CI coefficients fixed, the program evaluates the energy and the Fock matrix and adjusts the basis set coefficients describing the GVB natural orbitals accordingly, in basically the same manner used for the usual HF treatment. The revised orbitals and CI coefficients are then used in the next SCF iteration, and the process continues until both the GVB natural orbitals and the CI coefficients have converged.

The GVB treatment can also be applied to open shell cases, or restricted to certain electron pairs. These variations are described in reference [20], which also provides much more detail about the GVB methods and equations. The ability to restrict the use of GVB to particular electron pairs is an important strength of the method. This feature allows computationally inexpensive correlation of critical regions in very large molecules.

# 7.3. GVB-RCI WAVEFUNCTIONS

The GVB-RCI (restricted configuration interaction) wavefunction is the simplest multideterminantal reference wavefunction which properly dissociates to open shell fragments regardless of the spin multiplicity of the fragments. A critical advantage of GVB-RCI is that the GVB and RCI computations can be confined to a localized region of the molecule. The GVB-RCI method is therefore particularly useful for evaluating bond energies and bond formation and breaking, as well as for studies of open shell radicals and other systems for which it is important to avoid spin contamination problems.

The version of GVB-RCI within Jaguar uses pseudospectral numerical methods and a novel internal contraction scheme in which a GVB-PP wavefunction is used as a correlated mean field reference state [12]. This implementation of GVB-RCI can be used to generate highly accurate GVB-RCI wavefunctions, with energies within about 0.1 kcal/mole of results from all-analytical integral calculations [12]. The internal contraction scheme used restricts the number of CI coefficients in the RCI calculation to ~$n^3$, where n is the number of GVB pairs, yet is in excellent agreement with a fully uncontracted CI which by contrast would contain $2^n n^3$ CI coefficients (the number of uncontracted determinants).

The GVB-RCI program within Jaguar generates a correlated wavefunction from intra-pair excitations of the GVB reference wavefunction described in Section 7.2, using a highly effective contraction procedure to reduce the length of the CI expansions. The program employs the pseudospectral method to speed up integral evaluation, and systematically includes the most important configurations to make the calculation more practical, with minimal loss of accuracy relative to the fully uncontracted expansion.

The spatial states for an RCI pair are constructed from the same natural orbitals as those used for the GVB reference wavefunction, $\psi_{pg}$ and $\psi_{pu}$, but in addition to the GVB spatial state from Equation (6), rewritten here:

$$\xi_{p0} = C_{pg}\psi_{pg}^2 - C_{pu}\psi_{pu}^2 \tag{14a}$$

the RCI spatial states include the orthogonal complements $\xi_{p1}$ and $\xi_{p2}$:

$$\xi_{p1} = \psi_{pg}\psi_{pu} \tag{14b}$$

$$\xi_{p2} = C_{pu}\psi_{pg}^2 + C_{pg}\psi_{pu}^2 \tag{14c}$$

Just as the GVB method allows the user to correlate particular electron pairs for maximal efficiency, the RCI treatment can be applied to any user-specified subset of the GVB pairs. We then use a GVB mean field procedure to evaluate a Coulomb-exchange mean field operator describing the effect of the non-excited GVB pairs on the RCI pairs. This treatment effectively reduces the two-electron part of the Hamiltonian to the space of the RCI coordinates. Even for cases with many RCI pairs, we restrict the configurations to those with only a small number of excitations and use the mean field treatment for each configuration's calculation.

The RCI spatial states $\xi_{p1}$ add an extra complication to the necessary evaluation of Coulomb and exchange matrix elements using the natural orbitals $\psi_{pg}$ and $\psi_{pu}$. For the GVB case, it is sufficient to compute the following matrix elements (corresponding to Equation (10b) and Equation (10c)):

$$\left.\begin{aligned} J_{\nu\nu}^{\mu\mu} &= (\mu\mu|\nu\nu) = \left(\mu(1)\mu(1)\left|\frac{1}{r_{12}}\right|\nu(2)\nu(2)\right) \\ K_{\mu\nu}^{\mu\nu} &= (\mu\nu|\mu\nu) = \left(\mu(1)\nu(1)\left|\frac{1}{r_{12}}\right|\mu(2)\nu(2)\right) \end{aligned}\right\} \begin{array}{l}\mu \in \{\psi_{pg}, \psi_{pu}\}, \\ \nu \in \{\psi_{qg}, \psi_{qu}\}\end{array}$$

where the $\mu$ and $\nu$ can each be any natural orbital. For the RCI pairs, on the other hand, we need all matrix elements of the form:

$$\left.\begin{aligned} J_{\gamma\delta}^{\alpha\beta} &= (\alpha\beta|\gamma\delta) \\ K_{\beta\delta}^{\alpha\gamma} &= (\alpha\gamma|\beta\delta) \end{aligned}\right\} \quad (\alpha, \beta \in \{\psi_{pg}, \psi_{pu}\}, \gamma, \delta \in \{\psi_{qg}, \psi_{qu}\}),$$

where the $\alpha$ and $\beta$ natural orbitals are from the same RCI pair $p$ (and may be the same natural orbital), while the $\gamma$ and $\delta$ natural orbitals are from the same RCI pair with index $q$. The complicated part of the calculation of the Coulomb and exchange operators, then, is evaluating matrix elements in atomic orbital (AO) space and using the AO-

space matrix elements to produce the matrix elements in the natural orbital space, a process that normally requires a four-index transform.

By using the pseudospectral method, however, Jaguar reduces the scaling of the evaluation of each Coulomb or exchange matrix operator in basis function space from $N^4$ to $N^3$, and solves for the necessary matrix elements with a two-index transform rather than an expensive four-index one. For simplicity, we will describe this process for the Coulomb matrix elements only; the equations for $\boldsymbol{K}$ are similar. We first evaluate the usual three-center, one-electron integrals $A_{klg}$ in basis function space (see Equation (2)). We then evaluate $J_{\gamma\delta}$ in *physical* space for all $\gamma\delta$ corresponding to orbital products of each RCI pair, $\psi_{pg}\psi_{pg}$, $\psi_{pg}\psi_{pu}$, and $\psi_{pu}\psi_{pu}$, using the equation

$$J_{\gamma\delta g} = \sum_{kl} c_{k\gamma} c_{l\delta} A_{klg} \tag{15}$$

Next, we transform back into spectral space to get $J^{ij}_{\gamma\delta}$, where i and j are basis functions, using the pseudospectral method in the usual manner described in Section 7.1:

$$I^{ij}_{\gamma\delta} = \langle i|J_{\gamma\delta}|j\rangle = Q_{ig} J_{\gamma\delta g} R_{j\xi} \tag{16}$$

where $\boldsymbol{Q}$ is the pseudospectral least-squares operator and $R_{jg}$ is the value of the basis function $j$ at grid point $g$. We perform a final two-index transform,

$$J^{\alpha\beta}_{\gamma\delta} = \sum_{ij} c_{i\alpha} c_{j\beta} J^{ij}_{\gamma\delta} \tag{17}$$

to obtain the matrix elements in the natural orbital basis.

When Jaguar has obtained all Coulomb and exchange operators, it performs an iterative diagonalization of the Hamiltonian to obtain the RCI coefficients. We employ the Davidson method for this step.

# 7.4. Pseudospectral Local MP2 Techniques

Second order Møller-Plesset perturbation theory (MP2) is perhaps the most widely used ab initio electron correlation methodology, recovering a large fraction of the correlation energy at a relatively low

computational cost. The method greatly improves Hartree-Fock treatments of properties such as transition states, dispersion interactions, hydrogen bonding, and conformational energies. However, the scaling of conventional MP2 algorithms with system size is formally $nN^4$, where N is the number of basis functions and n the number of occupied orbitals, due to the necessity of carrying out a four index transform from atomic basis functions to molecular orbitals. In principle, it is possible to reduce this scaling by using integral cutoffs, as for Hartree-Fock calculations. However, the reduction is noticeably less effective in MP2, particularly for the large, correlation-consistent basis sets that are required for accurate correlation effects on observable quantities. Thus, MP2 techniques have traditionally been used primarily for small molecules.

Several years ago, Pulay and coworkers [35, 36] formulated a version of MP2 in which the occupied orbitals are first localized (e.g., via Boys localization [38]) and the virtual space correlating such orbitals are then truncated to a local space, built from the atomic basis functions on the local atomic centers orthogonalized to the occupied space. Another critical advantage of LMP2 (as for other localized correlation methods such as GVB and GVB-RCI) is that one can very precisely control which region of the molecule is correlated, reducing CPU costs enormously. The method has been shown to yield an accuracy for relative energies that is, if anything, superior to conventional MP2, due to elimination of basis set superposition error [37]. However, localized MP2 implementations in conventional electronic structure codes have not yet led to substantial reductions in CPU time, since the first few steps of the necessary four-index transform are unaffected by localization of the occupied orbitals, and the localized orbitals have tails that extend throughout the molecule.

We have carried out extensive tests demonstrating the accuracy and computational efficiency of the pseudospectral implementation of LMP2, as detailed in ref. [16]. In the pseudospectral approach, we assemble two-electron integrals over molecular orbitals directly and are thus able to fully profit from the huge reduction in the size of the virtual space in Pulay's theory. Formally, the PS implementation of LMP2 scales as $nN^3$; however, various types of cutoffs and multigrid procedures can reduce this to $\sim N^2$. In fact, for calculations involving both the 6-31G** and Dunning cc-pVTZ basis sets, we find a scaling $\sim N^{2.7}$ with system size.

The physical idea behind the LMP2 method is that if the molecular orbitals are transformed so that they are localized on bonds or electron pairs, correlation among the occupied pairs can be described by the local orbital pairs and their respective local pair virtual spaces

defined from the atomic orbitals on the relevant atom or pair of atoms. The localized orbitals can be generated by any unitary transformation of the canonical orbitals. For LMP2, we use Boys-localized [38] orbitals, for which the term $\Sigma_{ij}|\langle\varphi_i|r|\varphi_i\rangle - \langle\varphi_j|r|\varphi_j\rangle|^2$ is maximized. The local virtual space for each atom is defined by orthogonalizing its atomic basis functions against the localized molecular orbitals. The correlating orbitals included in the local virtual space are thus mostly near the atom itself, but because of the orthogonalization procedure, they are not particularly well localized.

The Jaguar LMP2 program uses Pulay's method [35, 36, 37] to expand the first order wavefunction correction $\Psi^{(1)}$ as a linear combination of determinants formed by exciting electrons from localized orbitals $i$ and $j$ to local virtual space correlation orbitals $p$ and $q$:

$$\Psi^{(1)} = \sum_{i \geq j} \sum_{pq} C_{ij}^{pq} \Psi_{ij}^{pq} \tag{18}$$

For local MP2, we must iteratively solve the following equation, which has been derived in detail by Pulay and Sæbo, for the coefficients $C_{ij}^{pq}$:

$$T_{ij}^{(2)} = K_{ij} + FC_{ij}S + SC_{ij}F - S\left(\sum_k \left[ F_{ik}C_{kj} + F_{kj}C_{ik} \right]\right)S = 0 \tag{19}$$

Here $\boldsymbol{F}$ is the Fock matrix and $\boldsymbol{S}$ is the overlap matrix. The exchange matrix $K_{ij}$ is restricted to the dimensions of the virtual space corresponding to the occupied localized molecular orbitals $i$ and $j$. The simplest updating scheme for the $C_{ij}$ coefficients is to obtain updated coefficients $C_{ij}'$ iteratively from the equation:

$$(C_{ij}^{pq})' = C_{ij}^{pq} + \frac{T_{ij}^{pq}}{\varepsilon_i + \varepsilon_j - \varepsilon_p^* - \varepsilon_q^*} \tag{20}$$

where $\varepsilon_i$ and $\varepsilon_j$ are the matrix elements $F_{ii}$ and $F_{jj}$ in the localized molecular orbital basis and $\varepsilon_p$ and $\varepsilon_q$ are the eigenvalues of the Fock matrix in the local virtual basis.

From the $C_{ij}$ coefficients and the exchange matrices $K_{ij}$, Jaguar computes the second order energy correction $E^{(2)}$ from the equations:

$$E^{(2)} = \sum_{i \geq j} \langle K_{ij} \tilde{C}_{ji} \rangle \tag{21a}$$

$$\tilde{C}_{ji} = (1 + \delta_{ij})^{-1}(4C_{ij} - 2C_{ji}) \tag{21b}$$

where the bracket in Equation (21a) denotes a trace and $\delta_{ij}$ is 1 if $i = j$ and 0 otherwise. Computing the exchange matrix elements for Equation (21a) is approximately 80% of the work for an energy correction computation, while generating the $C_{ij}$ coefficients comprises about 20% of the work.

Jaguar performs localized MP2 calculations using pseudospectral methods, evaluating integrals over grid points in physical space in a manner similar to that described for HF and GVB calculations in Section 7.1 and Section 7.2. The two-electron exchange integrals needed for Equation (21a) are evaluated over grid points $g$ as follows:

$$K_{ij}^{pq} = \sum_{g} Q_{ig} A_{jqg} R_{pg} \tag{22}$$

where $Q_{ig}$ is the least squares fitting operator for molecular orbital $i$ on grid point $g$, $R_{pg}$ is the physical space representation of virtual orbital $p$, and $A_{jqg}$ is the three-center, one-electron integral over the occupied molecular orbital $j$ and the local virtual orbital $q$. The last term is related to the three-center, one-electron integrals in atomic orbital space, $A_{klg}$, described in Equation (2), by

$$A_{jqg} = \sum_{kl} c_{kj} c_{lq} A_{klg} \tag{23}$$

Therefore, obtaining the exchange integral values requires solving for the integrals $A_{klg}$ in atomic orbital space; transforming these as:

$$A_{jlg} = \sum_{k} c_{kj} A_{klg} \; ; \tag{24}$$

performing the second transformation step to yield the integrals in molecular orbital space:

$$A_{jqg} = \sum_{l} c_{lq} A_{jlg} \; ; \tag{25}$$

and forming the necessary $K_{ij}^{pq}$ elements using Equation (22). Jaguar's local MP2 module also includes analytical corrections similar to those described earlier for Hartree-Fock and GVB calculations, and a length scales algorithm, both of which are explained in ref. [13].

# 7.5. DENSITY FUNCTIONAL THEORY

Density functional theory (DFT) is based on the Hohenberg-Kohn theorem [92], which states that the exact energy of a system can be expressed as a functional depending only on the electron density. In the Kohn-Sham implementation of DFT [93], this density is expressed in terms of Kohn-Sham orbitals $\{\psi_i\}$:

$$\rho(\vec{r}) = 2\sum_{i}^{\text{occ.}} |\psi_i(\vec{r})|^2 \tag{26}$$

similarly to the density expression used for Hartree-Fock SCF calculations. (For simplicity, we consider only closed shell systems in this overview of the method.)

The Kohn-Sham orbitals are expressed as a linear combination of basis functions $\chi_i(\vec{r})$ , and the coefficients for this expansion are solved iteratively using a self-consistent field method, as for Hartree-Fock. However, DFT includes exchange and/or correlation density functionals within the Fock matrix used for the SCF procedure. For DFT calculations, the Hartree-Fock exchange term $K_{ij}$ in the Fock matrix is replaced by the exchange-correlation potential matrix elements $V_{ij}^{xc}$ :

$$V_{ij}^{xc} = \int d\vec{r}\left(\left(\frac{\partial f_{xc}[\rho,\nabla\rho]}{\partial\rho}\right)\chi_i(\vec{r})\chi_j(\vec{r}) + 2\frac{\partial f_{xc}[\rho,\nabla\rho]}{\partial\gamma}\nabla\cdot(\chi_i(\vec{r})\chi_j(\vec{r}))\right) \tag{27}$$

where $f_{xc}[\rho,\nabla\rho]$ is an exchange-correlation functional and $\gamma$ is $\sqrt{\nabla\rho\cdot\nabla\rho}$ .

The exchange-correlation functional $f_{xc}[\rho,\nabla\rho]$ is usually separated into exchange and correlation functional components that are local or non-local in the density:

$$f_{xc}[\rho,\nabla\rho] = f_x[\rho] + f_{x,NL}[\rho,\nabla\rho] + f_c[\rho] + f_{c,NL}[\rho,\nabla\rho] \quad (28)$$

Under the local density approximation (LDA), the non-local functionals $f_{x,NL}[\rho,\nabla\rho]$ and $f_{c,NL}[\rho,\nabla\rho]$ are ignored; when either or both of these terms are included, the generalized gradient approximation (GGA), also known as the non-local density approximation (NLDA), applies. The local and non-local exchange and correlation functionals available within Jaguar are described in section 3.1 and its references.

The electronic ground state energy $E_0$ is given by

$$E_0[\{\psi_i\}] = 2\sum_i \int d\vec{r}\,\psi_i\left(-\frac{1}{2}\nabla^2\right)\psi_i + \int d\vec{r}\,V_{nuc}(\vec{r})\rho(\vec{r}) + \frac{1}{2}\int d\vec{r}\,J(\vec{r})\rho(\vec{r}) \quad (29)$$

(in Hartree atomic units), where $\boldsymbol{V}_{nuc}$ is the nuclear potential and $\boldsymbol{J}$ is the Coulomb potential. Therefore, for a given exchange-correlation functional, it is possible to solve iteratively for Kohn-Sham orbitals $\psi_i(\vec{r})$ and the resulting density $\rho$ to yield a final DFT energy.

A more detailed description of density functional theory can be found in references [94] and [95].

# 8. The Jaguar Input File

This chapter describes the Jaguar input file and how to use it to run Jaguar "by hand," that is, not using the interface. You might want to run Jaguar by hand in order to submit a job at a later time when computers are less busy, to use batch scripts to run multiple jobs in succession, to submit jobs from a non-X terminal, or to automate job submission with input files created by using other programs or by creating and editing input files yourself.

The first section of this chapter describes how to use the jaguar command to submit a single Jaguar job. Most of the rest of the sections in this chapter discuss the Jaguar input file format, describing the general file format first, then describing each section of the input file, starting with the geometry input (**zmat**) and the keyword (**gen**) sections. The final section, Section 8.20, describes how to run multiple Jaguar jobs using Jaguar's batch script.

## 8.1. THE jaguar COMMAND

As you already know, you can use the jaguar command to start Jaguar's graphical user interface. You can also use it to

- run a job on any machine at your site, with any version of Jaguar installed in the Jaguar home directory;
- kill a Jaguar job (if you were the one who started it) on any machine at your site;
- get a list of machines where Jaguar is installed; and
- get a list of jobs running on a particular machine.

If Jaguar is installed on more than one machine at your site, you can use the jaguar command on one machine to run, kill, or list Jaguar jobs on another machine, even if you are not logged in to the second machine.

The output from the command

```
jaguar help
```

summarizes how to use the jaguar command. This section describes in more detail how to use the jaguar command.

## *Selecting a Calculation Host*

If Jaguar is installed on several machines at your site, you can use the jaguar command to help determine which host you should use to run your job.

If you don't know which local machines have Jaguar installed, you can find out by entering the command

```
jaguar hosts
```

The hosts listed will be those in the jaguar.hosts file being used by the jaguar command. If you find that the list of hosts in the "jaguar hosts" output is incomplete, you probably need to edit the jaguar.hosts file indicated on the first line of the "jaguar hosts" output. See section 6.1 for a description of the jaguar.hosts file.

If a machine is often heavily used, you may want to check how many Jaguar jobs are running on it already before you submit yours. If perl is installed on your system (contact Schrödinger or your system manager if you need more information), you can do so with the command

```
jaguar jobs
```

for the machine on which you are logged in, or

```
jaguar jobs -h hostname
```

for another machine, where you substitute a host listed in the "jaguar hosts" output for *hostname*.

## *Selecting Particular Jaguar Executables*

By default, Jaguar looks for the executables available for the machine upon which you want to run a Jaguar job, then uses the most recent Jaguar executables for that machine type. However, if you have several differing sets of Jaguar executables at your site, such as different versions of Jaguar or executables for different machine types, you can choose to run your Jaguar job with a non-default choice of executables.

If you don't know which sets of Jaguar executables are available, you can find out by entering the command

```
jaguar versions
```

to find out about executables on the current host, or

```
jaguar versions -h hostname
```

to find out about executables on another machine (where you substitute a host listed in the "jaguar hosts" output for *hostname*).

## *Running a Jaguar Job from the Command Line*

The "jaguar run" command lets you run a Jaguar job using the Jaguar input file you specify. When you use the interface's Run window to select a calculation host and start a Jaguar job running on that machine, the interface starts the job by submitting the appropriate "jaguar run" command. If you want to, you can bypass the interface and submit the "jaguar run" command from your X terminal window instead, using any of the "jaguar run" command options shown in Table 8.1.1 and described below.

To run a Jaguar job, you first need an appropriate Jaguar input file. The file should be named in the form *jobname*.in, where *jobname* is whatever name you like. You can either create and edit an input file using your text editor, or you can use the interface to create and save one. See the subsection *Saving Input Files* in Section 2.7 for more information on using the interface to save input files. If you are hand-editing an input file, make sure its format agrees with that described in the rest of this chapter, especially Section 8.2.

You can run a Jaguar job from the command line with the command

```
jaguar run jobname
```

***Table 8.1.1.*** *Options for the "jaguar run" command.*

| option | effect | default behavior |
|---|---|---|
| **-h** | run Jaguar job on the designated host | run Jaguar job on the current host |
| **-w** | run the Jaguar job in the foreground (wait for job to finish before running next job) | run the Jaguar job in the background |
| **-v** | run the Jaguar executables that are in a directory whose path includes the string indicated | run the most recent version of Jaguar installed for the appropriate machine type |
| **-s** | temporary files and temp directory for job are saved at end of job | temporary files are cleaned out of temp directory and temporary directory is removed at end of job |
| **-p #** | use # processors for a parallel job | run a serial job |
| **-n** | Jaguar executables are run with nice -19 | Jaguar executables are run without nice |
| **-F** | force overwrite of scratch directory if it exists | abort if a scratch directory named for the job name already exists |
| **-t** | time stamps to log filer are done after each executable | time stamps to log file are done only at start and end of job |

where *jobname* reflects the name of your input file, *jobname*.in. With this command, the job will run on the machine upon which you have submitted the command, and will use the most recent version of Jaguar.

To run a Jaguar job on another machine, use a command in this form:

```
jaguar run -h hostname jobname
```

where your input file is named *jobname*.in and *hostname* is one of the hosts listed in the output from "jaguar hosts". For instance, if you were logged into a machine named alpha and wanted to run a job named ch4 on a machine named beta, you would enter

```
jaguar run -h beta ch4
```

To run a Jaguar job on the machine *hostname* with a particular, non-default set of executables, you can use the command

```
jaguar jobs -h hostname -v version
```

where *version* is *any* string that appears in one of the executable directories listed for that host by the "jaguar versions -h *hostname*" command.

The "jaguar run" command has several other command line options, as shown in Table 8.1.1. For example,

```
jaguar run -n -s jobname
```

causes executables to be run with a lower CPU scheduling priority (see man page on "nice") and leaves all temporary files generated during the job in the temporary directory.

If you wish to submit a series of jobs, you can create multiple input files using the Save window in the interface, as described in Section 2.7. You can then write a simple script including jaguar commands like the ones above to run the jobs, or run the jobs with the "jaguar batch" command. Section 8.20 describes how to run multiple jobs with the "jaguar batch" command.

## *Killing a Jaguar Job*

The "jaguar kill" command lets you kill any Jaguar job you are running on any local machine. This command is only available if perl is installed on your system; contact Schrödinger or your system manager if you need more information on perl.

Note that when you use the "jaguar kill" command, the temporary directory for your job will still exist and will contain *all* files generated during the job, and no output files will be copied back to your output directory.

To kill one of your Jaguar jobs, first enter the command

```
jaguar kill
```

to kill a job on the machine on which you are logged in, or

```
jaguar kill -h hostname
```

to kill a job running on another machine (where *hostname* is one of the hosts listed in the output from "jaguar hosts"). Your Jaguar jobs on the specified machine will be listed, and you will be prompted to

enter the number corresponding to the one you want to kill. To avoid killing any jobs, just hit <return>.

# 8.2. GENERAL DESCRIPTION OF THE INPUT FILE

The input file often begins with an optional line indicating the version number of Jaguar, such as "inv0300". The other parts of the input file are either single lines composed of flags in capital letters followed by arguments on the same line; sections describing the molecule and the calculation, whose formats will be described later in this chapter; or comments.

The input file should have the following format, where "[ ... ]" symbols denote optional entries, and entries in italics represent a character string with no spaces:

```
[inv0300]
[JOB: jobname]
[DIR: executable-directory temp-directory]
[comments]
{sections describing molecule & calculation}
[BASISFILE:file-path/name.basis]
[ATOMIGFILE:file-path/name.atomig]
[DAFFILE:file-path/name.daf]
[GRIDFILE:file-path/name.grid]
[CUTOFFFILE:file-path/name.cutoff]
[LEWISFILE:file-path/name.lewis]
[GPTSFILE:file-path/name]
```

The first three lines were all necessary in previous versions of Jaguar, but are no longer used or required, and the last six lines are only rarely used. Therefore, generally, your Jaguar input files will take a form as simple as

```
{sections describing molecule & calculation}
```

where only the **zmat** section, which contains the geometry and will be described later in this chapter, is actually required.

The .basis, .atomig (initial guess information), .daf (dealiasing functions), .grid, .cutoff, and .lewis data files are described in Chapter 9. If you want to use non-default choices for any of these files, you can specify their paths and names on the appropriate lines of the input file. If a file name listed in the input file ends with '.Z' (for example, "BASISFILE: erwin.basis.Z"), Jaguar will copy the file and uncompress it. You can specify a file on another host, or under another

account name on that host, by listing the file name in the format <host>:<file path and name on host>, or <user>@<host>:<file path and name on host>.

The GPTSFILE line allows you to use grid points and weights from an input file for any one grid used during the calculation. The file should have a line for each grid point, and each line should list, in order, the x, y, and z Cartesian coordinates (in angstroms) and the weight for that grid point. Grid weights are only used in charge fitting, so if you don't want to use them, use a '0' as a placeholder. For information about how to use this grid in a Jaguar calculation, see the *Grid and Dealiasing Function Keywords* subsection of Section 8.6.

Comments in the input file are ignored by Jaguar. If an input file was produced using the interface, text entered in the box marked Comment in the Run or Save window generally appears on the fourth line of the input file. If the geometry was symmetrized, as described in Section , a comment indicating the point group to which it was symmetrized appears.

## *Sections Describing the Molecule and Calculation*

The rest of the input file is composed of named sections. The sections may appear in any order. Character case (upper or lower) is ignored; therefore, either case, or a combination of the two, may be used. Equals signs ('='), commas (','), blank spaces (' '), and tabs are all considered spacing characters; however, if you plan to use the interface at all, we suggest that you use equals signs between a keyword and its value, and avoid using them anywhere else. Blank lines, or multiple spacing characters in a row, are equivalent to a single spacing character and thus may be used to improve readability.

The **gen** section contains a list of the general keywords which control the calculation. Defaults are provided for all unspecified keywords. The other sections contain lists, such as atomic coordinates. The sections currently allowed are shown in Table 8.2.1. Each section has

***Table 8.2.1.*** *Sections for Jaguar input files. (Many files contain only a few of these sections.)*

| section | description |
|---|---|
| **zmat** | Contains list of atomic coordinates describing molecular geometry, in Cartesian or Z-matrix format. |
| **zvar** | Sets values for **zmat** section variables. |

***Table 8.2.1.*** *(Continued) Sections for Jaguar input files. (Many files contain only a few of these sections.)*

| section | description |
|---|---|
| **coord** | Specify particular internal coordinates to be used for optimization. |
| **connect** | Specify particular internal coordinates to be used when generating coordinates for optimization. |
| **gen** | Sets general control keywords, including those describing the calculation performed, the grids, dealiasing functions, and cutoff parameters used, the electrostatic, geometry, and solvation properties calculated and the parameters used, and the output generated. |
| **gvb** | Sets GVB pairs. |
| **lmp2** | Sets LMP2 pairs for local local MP2 calculations, and/or delocalization of LMP2 pairs. |
| **atomic** | Sets atom-specific properties, including atomic masses (for isotopes), van der Waals radii for solvation calculations, and basis functions for individual atoms |
| **hess** | Allows input of initial nuclear Hessian. |
| **guess** | Allows input of initial wavefunction. |
| **pointch** | Adds independent point charges. |
| **efields** | Adds electric field or fields. |
| **ham** | Allows user input of Hamiltonian. |
| **orbman** | Allows orbitals to be reordered or linearly combined. |
| **echo** | One-word section indicating that the input file should be echoed in the output file. |
| **path** | Specifies execution path, listing order of Jaguar programs to be run. |
| **plot** | Allows data to be generated for plotting of orbital, potential, or densities. |
| **nbo** | Requests NBO (Natural Bond Orbital) calculation. |

a distinct format; the formats are described in detail in the rest of this chapter. Keywords in the **gen** section can have integer, real, or character string values. Generally, valid integer values are limited to a small set which differs for each keyword. Real values can optionally

include a 'd' or 'e' floating point power of ten. Character string keyword values may be limited to a small set, as for a basis set description, or may allow a general string like a file name.

Each section is delineated by a pair of '&' or '$' characters. The section name follows immediately after the first '&' or '$'. Thus, for example, the general keyword section may begin with '&gen' or '$gen' and ends with '&' or '$'. Within the **gen** section, allowed keywords are followed by numerical arguments giving their values, whose meanings are explained in Section 8.6. At least one spacing character must precede and follow each keyword and each value. For example,

```
&gen iguess=0 molchg=1 &
```

sets the **iguess** and **molchg** keywords of the **gen** section to 0 and 1, respectively. Sections may span multiple lines, and more than one section may appear in a line. However, a **gen** section keyword and its value must be on the same line. Note that the following example is interpreted in the same way as the **gen** section example given above:

```
This is a comment.
&gen iguess=0
   molchg=1 &
This is also a comment.
```

# 8.3. THE zmat, zmat2, AND zmat3 SECTIONS

The molecular geometry must be described in the **zmat** section. Details on inputting a geometry through the interface can be found in Section 2.2 and Section 2.4. The units for the geometry are set by the **iunit** keyword of the **gen** section; by default, these units are Angstroms and degrees.

If the geometry is in Cartesian coordinates, each line must contain four items: an atom name and the (x,y,z) coordinates. Each item should have at most 80 characters. The atomic label should begin with the one- or two-letter elemental symbol, in either uppercase or lowercase characters. Additional alphanumeric characters may be added, as long as the atomic symbol remains clear—for instance, "HE5" would be interpreted as helium atom "5", not hydrogen atom "E5". No more than four characters of the atomic label are kept. A sample Cartesian **zmat** section for a water molecule is:

```
&zmat
O   0.000000    0.000000   -0.113502
H1  0.000000    0.753108    0.454006
H2  0.000000   -0.753108    0.454006
&
```

A Z-matrix style **zmat** section should begin with the "&zmat" (or "$zmat") label and end with a & or $ character, should not include a line defining any variables (which are set in the **zvar** section described in Section 8.4), and should not contain any comment lines, but otherwise should have the same format described in Section 2.2 in the subsections *Z-Matrix Format for Geometry Input*, *Variables and Dummy Atoms in Z-Matrix Input*, and *Constraining Z-Matrix Bond Lengths or Angles*. One additional, optional feature is also available from the input file: you can orient the molecule or system according to a label on the same line as the "&zmat" section label. This orientation label should begin with the word "orient," which is followed by an option in the form *ab, -ab, a-b,* or *-a-b,* where *a* and *b* are each either x, y, or z (for example, "&zmat orient x-y"). Jaguar then assumes the first atom in the Z-matrix is at the origin, the second is along the *a* axis (in the negative direction for *-a*), and the third atom is in the *ab* plane, in the quadrant determined by the positive or negative signs of *a* and *b.*

Z-matrix input is interpreted in the units specified by **iunit**. Section 2.2 also includes a description of how to specify bond length or angle constraints on the Z-matrix coordinates for geometry optimizations.

To perform counterpoise calculations, you can add counterpoise atoms, which have the usual basis functions for that element but include no nuclei or electrons, by placing an at sign ('@') after the appropriate atoms' labels. For example, to place sodium basis functions at the Cartesian coordinates (0.0, 0.0, 1.0), you could include the following line in a Cartesian input file:

```
Na1@    0.0   0.0   1.0
```

You can also input counterpoise atoms for Z-matrix format geometries.

Finally, if you are optimizing a molecular structure to obtain a minimum-energy structure or a transition state, you may wish to refine the Hessian used for the job. (See section 4.3 for information on the methods used for transition state optimizations, including Hessian refinement.) If you put an asterisk ('*') after a coordinate value, Jaguar will compute the gradient of the energy both at the original geometry and at a geometry for which the asterisk-marked coordinate has been changed slightly, and will use the results to refine the

initial Hessian to be used for the optimization. (To request refinement of a coordinate whose value is set using a variable, add an asterisk ('*') to the end of the variable setting in the **zvar** section line that defines the variables.) For instance, a job run with this zmat section:

```
&zmat
O1
H2   O1   1.1*
H3   O1   1.1*   H2   108.0*
&
```

that included Hessian refinement would use both O–H bonds and the H–O–H angle in the refinement.

Molecular symmetry or the use of variables, either of which may constrain several coordinate values to be equal to each other, can reduce the number of coordinates actually used for refinement. For instance, for the water input example shown above, only two coordinates will actually be refined (the O–H bond distance, which is the same for both bonds, and the H–O–H angle) if molecular symmetry is used for the job.

Certain types of transition state optimizations require that you enter two or three geometries (see section 4.3 for details). For these jobs, you can input the second and/or third geometries in the **zvar2** and **zvar3** sections. (Note that the order of atoms in the input must be the same as in the **zmat** section.) Alternatively, if the changing coordinates in the **zmat** section are set using variables, you can leave out the **zmat2** and **zmat3** sections and specify the second and third geometries by adding **zvar2** and **zvar3** sections, which will be used in combination with the **zmat** section to define the second and third geometries. See Section 8.4 for details.

# 8.4. THE zvar, zvar2, AND zvar3 SECTIONS

The **zvar** section should contain a list of equations setting the values of any variables in the geometry input in the **zmat** section, in the same units used for the **zmat** section. Here is a sample **zvar** section:

```
&zvar
ycoor=0.753108  zcoor=0.454006
&
```

For an optimization, to constrain (freeze) all bond lengths or angles set to a particular variable, you should add a '#' sign to the end of the **zvar** section equation setting that variable. Similarly, to request Hessian refinement of a coordinate whose value is determined by a

variable setting in the **zvar** section, just add an asterisk ('*') to the end of the equation that sets the variable value in the **zvar** section.

For example, the **zvar** section

```
&zvar
ycoor=0.753108#  zcoor=0.454006
&
```

would, if used in an optimization, freeze all ycoor values to be equal to 0.753108 during the job.

Certain types of transition state optimizations require that you enter two or three geometries (see section 4.3 for details). For these jobs, you can specify variables for the second and/or third geometries in the **zvar2** and **zvar3** sections. If no **zmat2** or **zmat3** sections exist, these variables are used in combination with the **zmat** section to define the second and third geometries.

# 8.5. THE coord AND connect SECTIONS

For some geometry or transition state optimizations, you may want to specify that the optimizer use particular internal coordinates. For example, if you study a bond-forming reaction, you can require Jaguar to use the bond in question as an internal coordinate even when the bond distance is very long. You also may want to generate your own list of internal coordinates for cases that involve multiple separate (unbonded) fragments.

It is often useful to specify internal coordinates for pairs of atoms that are on separate sections of a large floppy molecule, but are close to being in van-der-Waals contact. Otherwise, small changes in a torsional coordinate far away from these atoms can then lead to steep changes in the energy. Adding explicit coordinates for these non-bonded contacts makes it possible for the optimization algorithm to control their approach more effectively.

To control the internal coordinates used in an optimization, you should first make sure that Jaguar is going to generate internal coordinates for the job. Optimization jobs generate and use redundant internal coordinates unless you have set the keyword **intopt** in the **gen** section of your input file. (See the subsection *Geometry Optimization and Transition State Keywords* in Section 8.6 for more details.)

To specify that particular bonds or angles should be included in the internal coordinates generated and used for an optimization, use a

**coord** section. Each line of a **coord** section should contain a list of atoms used to specify a bond, bond angle, or torsional angle coordinate to be included among the internal coordinates generated by Jaguar. If you want to hold the coordinate fixed at its initial value throughout the job, add the entry '#' to the end of the line (after one or more spacing characters).

As an example, the **coord** section

```
&coord
C1  C2
C1  C2  C3 #
C1  C2  C3  C4
&
```

requests that the set of internal coordinates include the C1–C2 bond, the C1–C2–C3 bond angle (which is to be held frozen throughout the optimization), and the C1–C2–C3–C4 torsion.

Alternatively, you can use a **connect** section to specify the bonds used by Jaguar in its generation of internal coordinates. Each line of a **connect** section should list two atoms by either their atom labels (such as 'H2' for a hydrogen) or their atom numbers (such as '3' for the third atom listed in the **zmat** section input). Here is a sample **connect** section:

```
&connect
C1  C2
C2  C3
&
```

The two atoms on each line of the **connect** section are then treated as nearest neighbors by the program when it generates redundant internal coordinates for the optimization. Consequently, the internal coordinates generated by Jaguar include the bond between those two atoms and angles between those two atoms and any other atoms that are nearest neighbors to either of them. For the sample **connect** section above, for instance, the redundant internal coordinates would include the C1–C2 bond, the C2–C3 bond, and the C1–C2–C3 angle in addition to whatever internal coordinates would be generated without the **connect** section.

# 8.6. THE gen SECTION

The keywords of the **gen** section allow control over how the calculation is performed. Many of these keywords can be set from the interface. See Chapters 3 and 5 for details.

Throughout this section, the default values for keywords are indicated in bold italics. The keywords for geometry input are described first, followed by those relating to correlation methods, optimization to a minimum-energy structure or transition state, calculations in solution, calculation of various molecular properties, basis sets, SCF methods, and output. These subsections correspond to the order of information in Chapters 3 and 5. Finally, keywords relating to grids and dealiasing functions, cutoff parameters, and memory usage are described.

## *Geometry Input Keywords*

The keywords **iunit** and **covfac** help determine how the geometry input from the **zmat** section will be interpreted. The **iunit** keyword, whose default value is 1, describes what units the geometry is assumed to have, as indicated in Table 8.6.1.

***Table 8.6.1.*** *Options for the keyword* ***iunit****.*

| keyword | value | description |
|---|---|---|
| **iunit** | 0 | geometry units are bohr and radians |
| | ***1*** | geometry units are Angstroms and degrees |
| | 2 | geometry units are bohr and degrees |
| | 3 | geometry units are Angstroms and radians |

The real-valued keyword **covfac** determines which atoms are considered to be bonded. Two atoms are bonded if they are closer to each other than **covfac** times the sum of their covalent radii. The default value for this variable is 1.2.

## *Molecular State Keywords (Charge and Multiplicity)*

The keywords that describe the input molecule's charge and spin multiplicity are shown in Table 8.6.4. These keywords correspond to interface options described in Section 2.3.

## *Atomic Mass Keyword*

The keyword **massav** determines the atomic masses used for any atoms whose masses or isotopes are not specifically set in the **atomic** section (see Section 8.9). The masses used are from ref. [100].

***Table 8.6.2.*** *Keywords to describe the molecular state.*

| keyword | value | description |
|---|---|---|
| **molchg** | any | overall charge on molecule, excluding point charges set in **pointch** section (default=0) |
| **multip** | >0 | spin multiplicity: 1 for singlet, 2 for doublet, etc. (default=1, except for **ihamtyp**=0, when **multip**=2 by default) |

***Table 8.6.3.*** *Keyword to describe the atomic masses used.*

| keyword | value | description |
|---|---|---|
| **massav** | ***0*** | use masses of most abundant isotopes as atomic masses |
| | 1 | use average isotopic masses as atomic masses, where averages are weighted according to natural abundance of isotopes |

## *Symmetry-Related Keywords*

By default, for most calculations, Jaguar takes advantage molecular symmetry to reduce computing time, as described in Section . Several integer-valued keywords shown in Table 8.6.4 describe how the

***Table 8.6.4.*** *Symmetry-related keywords in Jaguar.*

| keyword | value | description |
|---|---|---|
| **isymm** | 0 | do not use symmetry |
| | 1 | rotate atomic grids to match molecular symmetry, if possible |
| | 2 | change grids to get molecular symmetry, if necessary |
| | 4 | use symmetry in preprocessing, but not in SCF |
| | ***8*** | use symmetry in preprocessing and SCF |

***Table 8.6.4.*** *(Continued) Symmetry-related keywords in Jaguar.*

| keyword | value | description |
|---|---|---|
| **ipopsym** | ***0*** | allow change in number of electrons in each irreducible representation (default for HF and DFT closed-shell jobs) |
| | 1 | don't allow number of electrons in each irreducible representation to change (default for non-HF, non-DFT and open-shell calculations) |
| **idoabe** | ***0*** | allow non-Abelian point group symmetry assignment |
| | 1 | allow only Abelian point group symmetry assignment |

program uses symmetry. Default values for these keywords are shown in bold italics.

## *GVB and Lewis Dot Structure Keywords*

The **ihfgvb** keyword allows you to specify the initial guess to be used for a generalized valence bond (GVB) calculation. By default, **ihfgvb** is set to 0. The **ihfgvb** keyword is described in this section under *Initial Guess Keywords*.

GVB pairs are set in the **gvb** section, where pairs to be used in an RCI (restricted configuration interaction) calculation are also specified, and a GVB calculation will be performed any time one or more GVB pairs are described in the input file.

You can find Lewis dot structures by setting the appropriate keywords, and you can also use one of these structures to set GVB pairs automatically. The appropriate keywords are listed in Table 8.6.5 with their default values indicated in bold italics.

The Lewis dot structure code finds several alternative Lewis dot structures for resonant molecules, assigning bonds as single, double, or triple bonds unambiguously. (For instance, it finds two structures for benzene, depending on the assignment of the pi bonds.) For these cases, you may wish to run Jaguar with **lewdot**=–1 and **lewstr**=0, which will cause it to print out all Lewis dot structures it finds, then exit. At that point, you can figure out which structure you want to use to set the GVB pairs, set **lewstr**, **igvball**, and **igvbsel** accordingly, and set **lewdot**=1.

***Table 8.6.5.*** *Keywords for evaluation of Lewis dot structures and application of that information to GVB pair settings.*

| keyword | value | description |
|---|---|---|
| **lewdot** | ***0*** | do not find Lewis dot structure(s) or use them to set GVB pairs |
| | 1 | find Lewis dot structure(s) and continue on with calculation (**lewdot**=1 by default if **igvball** > 0) |
| | –1 | find Lewis dot structure(s) and exit without performing SCF or other later calculations |
| **lewstr** | ***0*** | print all Lewis dot structures if **lewdot**=1 or –1 |
| | >0 | use structure number lewstr for output and/or setting GVB pairs (**lewstr**=1 by default if **igvball** > 0) |
| **igvball** | ***0*** | do not select any GVB pairs based on Lewis dot structure |
| | 1 | select GVB pairs for any atoms according to **igvbsel** and Lewis dot structure **lewstr** |
| | 2 | select heteroatom GVB pairs only, according to **igvbsel** and Lewis dot structure **lewstr** (heteroatom pairs are all pairs whose atoms are different elements, except for C–H pairs) |
| **igvbsel** | 1 | select only sigma GVB pairs |
| | 2 | select only pi and second pi GVB pairs |
| | 3 | select only sigma, pi, and second pi GVB pairs |
| | 4 | select only lone GVB pairs |
| | 5 | select only lone and sigma GVB pairs |
| | 6 | select only lone, pi, and second pi GVB pairs |
| | ***7*** | select sigma, pi, second pi, and lone GVB pairs (default when **igvball** > 0) |

If you know there is only one reasonable Lewis dot structure for the molecule, you can simply set **igvball** and **igvbsel**. At that point, **lewdot** and **lewstr** will be set to 1 by default.

The values for **igvbsel** are easier to remember if you associate the number 1 with sigma pairs, 2 with pi pairs, and 4 with GVB lone pairs. Then, to print out any combination of these pair types, you set **igvbsel** to equal the sum of the numbers associated with the pair types you wish to print.

## *LMP2 Keywords*

The **mp2** keyword allows you to request a local Møller-Plesset perturbation theory (LMP2) calculation. By default, LMP2 is off (**mp2**=0). For more information on the local MP2 method, see Section 3.2 and section 7.4

Please note that Jaguar LMP2 calculations require a basis set that allows the pseudospectral method to be used. See Table 3.8.1 and Table 3.8.2 in Section 3.8 to obtain this basis set information.

Local MP2 calculations use the LMP2 method for all atoms unless the **lmp2** section described in Section 8.8 is used to set local LMP2 pairs or unless the keyword **iheter** is set to 1. The **iheter** and **mp2** keyword settings are described in Table 8.6.6.

***Table 8.6.6.*** *Keyword settings for local MP2 calculations.*

| keyword | value | description |
|---|---|---|
| **mp2** | ***0*** | do not run local second-order Møller-Plesset perturbation theory (LMP2) calculation |
| | 1 | correlate core and valence electrons |
| | 3 | run LMP2 calculation (for valence electrons only) |
| **iheter** | ***0*** | treat all atoms with LMP2 if LMP2 is on unless **lmp2** section exists; if LMP2 is on and **lmp2** section exists, set atom pairs in **lmp2** section |
| | 1 | treat only heteroatom pairs (atoms in bonds with atoms of other elements, except C atoms bonded only to C and/or H) and any pairs set in **lmp2** section at LMP2 level, other atoms at HF level |

***Table 8.6.6.*** *(Continued) Keyword settings for local MP2 calculations.*

| keyword | value | description |
|---|---|---|
| **ireson** | ***0*** | do not delocalize LMP2 pairs over other atoms |
| | 1 | calculate Lewis dot structure of molecule (by setting **lewdot** = 1), then delocalize LMP2 pairs on any bond in an aromatic ring of <7 atoms over neighboring atoms in the aromatic ring |
| | 2 | calculate Lewis dot structure of molecule (by setting **lewdot** = 1), then delocalize LMP2 pairs on any bond in an aromatic ring of <7 atoms over all atoms in the aromatic ring |
| **idelocv** | ***0*** | do not delocalize any pairs listed in **lmp2** section (default for all calculations except those with **iqst**>0 and/or **ireson**>0) |
| | 1 | treat all LMP2 pairs, but delocalize any pairs in **lmp2** section as indicated there, or (default for QST-guided transition state searches) delocalize any pairs on atoms with breaking or forming bonds |
| | 2 | perform a local local MP2 calculation, treating only pairs listed in the **lmp2** section at the LMP2 level, and also delocalize any pairs in **lmp2** section as indicated there |
| **loclmp2c** | 0 | do not localize core orbitals for LMP2 calculation |
| | 1 | perform Boys localization on core orbitals for LMP2 calculation |
| | ***2*** | perform Pipek-Mezey localization on core orbitals for LMP2 calculation, maximizing Mulliken atomic populations |
| | 3 | perform Pipek-Mezey localization on core orbitals for LMP2 calculation, maximizing Mulliken basis function populations |

***Table 8.6.6.*** *(Continued) Keyword settings for local MP2 calculations.*

| keyword | value | description |
|---|---|---|
| **loclmp2v** | 1 | perform Boys localization on valence orbitals for LMP2 calculation |
| | ***2*** | perform Pipek-Mezey localization on valence orbitals for LMP2 calculation, maximizing Mulliken atomic populations |
| | 3 | perform Pipek-Mezey localization on valence orbitals for LMP2 calculation, maximizing Mulliken basis function populations |

For LMP2 calculations, Jaguar needs to obtain localized orbitals. By default, Jaguar uses the Pipek-Mezey method to perform the localization. If Pipek-Mezey localization does not converge for a particular case, you may wish to try Boys localization by changing the settings for the keywords **loclmp2c** and **loclmp2v**, as indicated in Table 8.6.6. If you are performing a set of calculations to compare against each other, you should use the same localization method for all of the calculations.

## *DFT Keywords*

To use density functional theory (DFT) for single point energies, geometry optimizations, or calculations of various other properties, you should set the **idft** keyword. If you wish to evaluate the (non-self-consistent) energy of the final, post-SCF wavefunction using a particular set of functionals, you can use the **jdft** keyword. Most DFT options described here are also available from the interface, as described in Section 3.1. For information on setting the keywords associated with grids for DFT calculations, please see the *Grid and Dealiasing Function Keywords* subsection near the end of this section.

Positive values of **idft** describe both the exchange and correlation functionals, if any. The value of **idft** can be broken down in the form **idft** = 10000****i*** + 1000****j*** + 100****k*** + 10****l*** + ***m***, or **idft** = ***ijklm***, where the values of ***j***, ***k***, ***l***, and ***m*** determine the exchange and correlation functionals and ***i*** specifies particular coefficients for the functionals. The functionals themselves are determined as described in Table 8.6.7 through Table 8.6.10.

For instance, if **idft**=1301, the DFT calculation employs the Slater local exchange functional and the Perdew-Zunger local correlation

***Table 8.6.7.*** *Values of **m** in **idft** (where **idft = ijklm**).*

| ***m* in idft** | **local exchange functional (or exact exchange)** |
|---|---|
| ***m***=0 | exact exchange (Hartree-Fock) |
| ***m***=1 | Slater |
| ***m***=9 | Xα |

***Table 8.6.8.*** *Values of **l** in **idft** (where **idft = ijklm**).*

| ***l* in idft** | **non-local exchange functional** |
|---|---|
| ***l***=0 | none |
| ***l***=1 | Becke, 1988 (non-local term only) |
| ***l***=4 | Perdew-Wang GGA-II, 1991 (non-local exchange only) |

***Table 8.6.9.*** *Values of **k** in **idft** (where **idft = ijklm**).*

| ***k* in idft** | **local correlation functional** |
|---|---|
| ***k***=0 | none |
| ***k***=1 | Vosko-Nusair-Wilk (VWN) |
| ***k***=2 | VWN5 |
| ***k***=3 | Perdew-Zunger, 1981 |
| ***k***=4 | Perdew-Wang GGA-II, 1991 (local correlation only) |

***Table 8.6.10.*** *Values of **j** in **idft** (where **idft = ijklm**).*

| ***j* in idft** | **non-local correlation functional** |
|---|---|
| ***j***=0 | none |
| ***j***=1 | Perdew, 1986 (non-local gradient correction) |
| ***j***=2 | Lee-Yang-Parr (local + non-local) |
| ***j***=4 | Perdew-Wang GGA-II, 1991 (non-local correlation only) |

functional with Perdew's 1986 non-local correlation functional. A typical local density approximation (LDA) calculation could use **idft**=101, while **idft**=2011 sets a popular NLDA choice typically called BLYP. If you specify the Lee-Yang-Parr functional, which contains local and non-local terms, you may not specify a local correlation functional (i.e. if ***j***=2, ***k*** must be 0) unless you are using the Becke 3 parameter hybrid method, as described below.

If the value of ***i*** within **idft** is 1 or 2, the functionals given by ***j***, ***k***, ***l***, and ***m*** are combined using coefficients determined by the appropriate hybrid method, as indicated in Table 8.6.11.

***Table 8.6.11.*** *Values of* ***i*** *in* ***idft*** *(where* ***idft = ijklm****).*

| ***i*** **in idft** | **hybrid method** |
|---|---|
| ***i***=0 | none |
| ***i***=1 | half & half (functional coefficients are all 0.5) |
| ***i***=2 | Becke 3 parameter (parameters from ref. [26]) |

For the half & half hybrids, half of the exact exchange is automatically included with half of the selected exchange functional. The coefficient of any local correlation functional or non-local exchange or correlation functional is also set to 0.5. You must specify a Slater or Xα local exchange functional for a half & half hybrid, and if you use the Lee-Yang-Parr functional, you may not specify a local correlation functional.

For Becke 3 parameter hybrids, you need to specify a Slater or Xα local exchange functional, a non-local exchange functional, a local correlation functional, and a non-local correlation functional (i.e. ***j***, ***k***, ***l***, and ***m*** must all be non-zero if ***i*** is 2). Even when you use the Lee-Yang-Parr functional in a Becke 3 parameter hybrid, you must list a purely local correlation functional, which will be used to adjust the local correlation contribution. For Becke 3 parameter hybrids that do not include the Lee-Yang-Parr functional, the coefficients of the exact HF exchange and of the local exchange, non-local exchange, local correlation, and non-local correlation functionals are 0.2, 0.8, 0.72, 1.0, and 0.81, respectively. If the Lee-Yang-Parr functional is used in a Becke 3 parameter hybrid, its coefficient is 0.81, and the coefficient of the local correlation functional used is 0.19.

If **idft**=−1, the values of the keywords **xhf**, **xexl1**, **xexl9**, **xexnl1**, and **xexnl4** determine the contributions of the exact exchange and various exchange functionals, while the keywords **xcorl1**, **xcorl2**, **xcorl3**, **xcorl4**, **xcornl1**, **xcornl2**, and **xcornl4** control the contributions of the correlation functionals, as listed in Table 8.6.12.

***Table 8.6.12.*** *Functional coefficient keywords.*

| **keyword** | **corresponding functional (or exact exchange)** |
|---|---|
| **xhf** | exact exchange (Hartree-Fock) |

***Table 8.6.12.*** *(Continued) Functional coefficient keywords.*

| keyword | corresponding functional (or exact exchange) |
|---|---|
| **xexl1** | Slater local exchange functional |
| **xexl9** | Xα local exchange functional |
| **xexnl1** | Becke, 1988 (non-local gradient correction) |
| **xexnl4** | Perdew-Wang GGA-II, 1991 non-local exchange functional |
| **xcorl1** | VWN local correlation functional |
| **xcorl2** | VWN5 local correlation functional |
| **xcorl3** | Perdew-Zunger 1981 local correlation functional |
| **xcorl4** | Perdew-Wang GGA-II, 1991 local correlation functional |
| **xcornl1** | Perdew 1986 non-local gradient correction |
| **xcornl2** | Lee-Yang-Parr local + non-local correlation functional |
| **xcornl4** | Perdew-Wang GGA-II, 1991 non-local correlation functional |

For example, with the keyword settings **idft**=–1, **xhf**=0.332, **xexl1**=0.575, and **xcorl1**=0.575, and with all other **xex** and **xcor** keywords set to zero, the exchange is treated with a combination of the exact exchange and the Slater local functional, while the correlation functional is pure VWN.

Finally, if you wish to evaluate the energy of the final, post-SCF wavefunction using a particular functional or combination of functionals, you should use the keyword **jdft**. This keyword can take on the same values as **idft**, and the meanings for each value are the same as those described above for **idft**. If **jdft**=–1, you can set up a customized functional using the keywords **yhf**, **yexl1**, **yexl9**, **yexnl1**, **yexnl4**, **ycorl1**, **ycorl2**, **ycorl3**, **ycorl4**, **ycornl1**, **ycornl2**, and **ycornl4**, which correspond to the keywords in Table 8.6.12 (e.g. **xexl1**). If you do a post-SCF DFT energy evaluation, you should not perform a geometry optimization or calculate the solvation energy, polarizability, or any other properties.

For any DFT job, the Methods keyword **vshift** is set to 0.2 (for hybrid methods) or 0.3 (for non-hybrid methods) by default, and the Methods keyword **idenavg** is set to 1 by default, in order to help convergence.

More complete descriptions and references for each DFT functional and hybrid are given in Section 3.1.

## *Geometry Optimization and Transition State Keywords*

Many of the keyword settings for optimization of minimum-energy structures and transition states described in this subsection can be made from the interface, as described in Chapter 4, which also contains more details about the methods used for optimizations.

Table 8.6.13 contains optimization keywords that take on integer values. Most default values for these integer keywords are indicated in bold italics, and only the values listed in the table are allowed. In cases where the default is different for optimizations to minimum-energy structures than it is for transition state optimizations, both defaults are in bold italics, and the cases for which each is a default are explained in the keyword description.

***Table 8.6.13.*** *Integer keywords for geometry and transition state optimization calculations.*

| keyword | value | description |
|---|---|---|
| **igeopt** | ***0*** | do not optimize molecular geometry |
| | 1 | optimize minimum-energy structure |
| | –1 | calculate forces but do not perform geometry optimization |
| | 2 | optimize transition state geometry |
| **iqst** | ***0*** | perform standard (non-QST) transition state search |
| | 1 | use quadratic synchronous transit (QST) methods to guide transition state search |
| **nogas** | ***0*** | for optimizations in solution, perform gas phase geometry optimization first (to get accurate solvation energy) |
| | 1 | for optimizations in solution, skip gas phase geometry optimization and compute solvation energies using **esolv0** value (from input file) as gas phase energy (should yield same *structure* as **nogas**=0) |

***Table 8.6.13.*** *(Continued) Integer keywords for geometry and transition state optimization calculations.*

| keyword | value | description |
|---|---|---|
| | 2 | for optimizations in solution, skip gas phase geometry optimization and compute solvation energies using energy of initial structure as gas phase energy (should yield same *structure* as **nogas**=0) |
| **intopt** | 0 | use Cartesian coordinates for optimization |
| | ***1*** | use internally generated internal coordinates for optimization (including any from **coord** or **connect** sections, if they exist) |
| | 2 | use internal coordinates from input Z-matrix for optimization (note: if geometry input is in Cartesian format or contains a second bond angle rather than a torsional angle for any atom, **intopt** is reset to 1) |
| **nmder** | ***0*** | if calculating forces, compute analytic derivatives of energy |
| | 1 | if calculating forces, compute numerical derivatives of energy (obtained from calculations on 6 $N_{atom}$ perturbed geometries by moving each atom **pertnd** bohr in positive or negative x, y, or z direction) |
| | 2 | calculate frequencies numerically |
| **maxitg** | >0 | maximum number of optimization iterations (maximum number of structures generated); default is ***100*** |
| **iaccg** | ***2*** | use default convergence criteria shown in Table 8.6.15 |
| | 3 | perform quicker, coarser calculation by multiplying convergence criteria shown in Table 8.6.15 by 5 |
| | 4 | solution-phase criteria; a factor of 3 times the criteria shown in Table 8.6.15 |

***Table 8.6.13.*** *(Continued) Integer keywords for geometry and transition state optimization calculations.*

| keyword | value | description |
|---|---|---|
| **nogdiis** | 0 | use GDIIS method (Geometry optimization by Direct Inversion in the Iterative Subspace) [99] to get new geometry |
| | ***1*** | don't use GDIIS method |
| **ilagr** | 0 | apply constraints by zeroing gradient along frozen coordinates (pre-v2.3 method; not recommended) |
| | ***1*** | apply constraints using Lagrange multipliers |
| **igrado** | ***0*** | use energy and gradient information in minimization and convergence evaluation |
| | 1 | use only gradient information in minimization and convergence evaluation |
| **nooptr** | ***0*** | optimize all bond lengths not specifically constrained in **zmat** section |
| | 1 | constrain (freeze) all bond lengths for optimization |
| **noopta** | ***0*** | optimize all bond angles not specifically constrained in **zmat** section |
| | 1 | constrain (freeze) all bond angles for optimization |
| **nooptt** | ***0*** | optimize all torsional angles not specifically constrained in **zmat** section |
| | 1 | constrain (freeze) all torsional angles for optimization |
| **inhess** | –1 | use Fischer-Almlöf guess for Hessian |
| | ***0*** | use Schlegel guess for Hessian (default choice only if no **hess** section exists) |
| | 1 | use unit matrix for initial Hessian |
| | 2 | use Cartesian input Hessian found in **hess** section (**inhess=**2 automatically if non-empty **hess** section exists) |
| | 4 | compute and use quantum mechanical Hessian |

***Table 8.6.13.*** *(Continued) Integer keywords for geometry and transition state optimization calculations.*

| keyword | value | description |
|---|---|---|
| **irefhup** | 2 | refine initial Hessian using Powell updates [101] |
| | 3 | refine initial Hessian using mixed Murtagh-Sargent/Powell updates [102] |
| | ***4*** | refine initial Hessian using Murtagh-Sargent updates [103] |
| **nhesref** | >0 | number of lowest-frequency Hessian eigenvectors used in Hessian refinement (default is ***0***) |
| **ihuptyp** | 0 | don't update Hessian |
| | ***1*** | update Hessian each iteration using BFGS (Broyden-Fletcher-Goldfarb-Shanno) method [104] (default for minimum-energy structure optimizations) |
| | 2 | update Hessian using Powell method [101] |
| | ***3*** | update Hessian using mixed Murtagh-Sargent/Powell method [102] (default for transition state optimizations) |
| | 4 | update Hessian using Murtagh-Sargent method [103] (not recommended) |
| | –1 | compute quantum mechanical Hessian at each geometry; sets **inhess**=4 |
| **irfo** | 0 | before using Hessian to update geometry, modify it by sign-flipping or reverting to an older Hessian [100] |
| | 1 | before using Hessian to update geometry, modify it by RFO (rational function optimization) level shifting [105] |
| | ***2*** | before using Hessian to update geometry, modify it by P-RFO (partitioned rational function optimization) level shifting [105] |

***Table 8.6.13.*** *(Continued) Integer keywords for geometry and transition state optimization calculations.*

| keyword | value | description |
|---|---|---|
| **ifollow** | ***0*** | for each transition state optimization iteration, select a new eigenvector to follow |
| | 1 | for each optimization iteration, follow eigenvector that most closely correlates with one followed previously |
| **itrvec** | ***0*** | for transition state optimization, select lowest Hessian eigenvector as transition vector |
| | >0 | select eigenvector number **itrvec** as transition vector (see section 4.3) |
| | –1 | select lowest non-torsional eigenvector as transition vector |
| | –2 | select lowest stretching eigenvector as transition vector |
| | -5 | select eigenvector which best represents reaction path |
| **itradj** | ***0*** | use same trust radius throughout optimization (default for minimum-energy structure optimizations) |
| | ***1*** | adjust trust radius using Culot/Fletcher heuristic [104, 106] (default for transition state optimizations) |
| | –1 | adjust trust radius using Simons' cubic potential model [107] (not recommended with Jaguar) |
| **itrcut** | 0 | apply trust radius by truncating Newton-Raphson step(s) |
| | ***1*** | apply trust radius by level shifting of Hessian to reduce resultant step size |

Several real-valued keywords also control optimizations. These keywords are listed in Table 8.6.14 and Table 8.6.15. Note that all keyword values must be greater than or equal to zero.

The real-valued keywords shown in Table 8.6.15 may be used to specify the geometry convergence criteria, or these criteria may be scaled to five times their default values with the keyword setting **iaccg**=3 for a quicker, coarser calculation. The first four real-valued

***Table 8.6.14.*** *Real-valued optimization keywords (except for convergence criteria, which are shown in Table 8.6.15). All values must be set greater than or equal to 0.*

| keyword | description |
|---|---|
| **pertnd** | displacement (in atomic units) used for Hessian refinement or calculations of numerical forces or frequencies; default is ***0.05*** |
| **trust** | initial trust radius, in atomic units (bohr and/or radians): if norm of proposed displacements exceeds trust radius, step size is reduced as described by **itrcut** (**trust** default is ***0.3***, except for solvated cases or transition state optimizations, when it is ***0.1***) |
| **tradmx** | maximum trust radius allowed during optimization for **itradj**>0; see **trust** information (**tradmx** default is ***0.3,*** except for solvated cases, when it is ***0.1***) |
| **tradmn** | minimum trust radius allowed during optimization for **itradj**>0; see **trust** information (**tradmn** default is ***0.01***) |
| **tremx** | trust radius reduction criterion; if relative error between actual and predicted energy changes is more than **tremx** and **itradj**>0, trust radius is reduced (default is ***0.25***) |
| **trgmx** | trust radius reduction criterion; for **itradj**>0 and **trgmx**>0, if absolute error in a component of predicted gradient exceeds **trgmx** Hartrees/bohr, trust radius is reduced (**trgmx**=***0.0*** by default) |
| **treok** | criterion for increasing trust radius; if **itradj**=2 and relative error between actual and predicted energy changes is less than **treok**, trust radius is increased (**treok** default is ***0.2***) |
| **trescal** | scale factor for trust radius adjustment (default is ***2.0***); used only when **itradj**=2 |

keywords listed in Table 8.6.15 have units of Hartrees/bohr, **gconv5** and **gconv6** have units of bohr, and **gconv7** has units of Hartrees.

***Table 8.6.15.*** *Geometry convergence criteria keywords.*

| keyword | default value | convergence criterion for: |
|---|---|---|
| **gconv1** | $4.5 \times 10^{-4}$ | maximum element of gradient |
| **gconv2** | $3.0 \times 10^{-4}$ | rms of gradient elements |

***Table 8.6.15.*** *(Continued) Geometry convergence criteria keywords.*

| keyword | default value | convergence criterion for: |
|---|---|---|
| **gconv3** | $1.0 \times 10^{-2}$ | maximum Newton-Raphson step (note: not currently used) |
| **gconv4** | $1.0 \times 10^{-2}$ | rms Newton-Raphson step (note: not currently used) |
| **gconv5** | $1.8 \times 10^{-3}$ | maximum element of nuclear displacement |
| **gconv6** | $1.2 \times 10^{-3}$ | rms of nuclear displacement elements |
| **gconv7** | $5.0 \times 10^{-5}$ | difference between final energies from previous and current geometry optimization iterations |

Please note also that SCF calculations performed for each new structure generated during an optimization are judged to be converged when they meet the criterion for the root mean square of the change in density matrix elements, which is controlled by the keyword **dconv**; the usual SCF energy convergence criterion (**econv**) is ignored for optimizations.

## *Solvation Keywords*

Most of the solvation keywords correspond to interface options described in Section 3.5. The allowed values for the integer solvation keywords are described in Table 8.6.16. Defaults for these keywords

***Table 8.6.16.*** *Integer keywords for solvation calculations.*

| keyword | value | description |
|---|---|---|
| **isolv** | 0 | do not perform a solvation calculation |
| | 2 | perform a solvation calculation using Jaguar's Poisson-Boltzmann solver |
| **icavity** | 0 | do not include solute cavity energy term in solvation calculation |
| | ***1*** | include solute cavity energy term (default when the solvent is water) |
| | 2 | force calculation of cavity energy term |

***Table 8.6.16.*** *(Continued) Integer keywords for solvation calculations.*

| keyword | value | description |
|---|---|---|
| **isurf** | 0 | do not include first shell correction factor term in solvation energy |
| | 1 | include first shell correction factor term in solvation energy (default for most calculations in water; turns on Lewis dot keyword **ivanset**=1 by default) |
| **ivanset** | 0 | do not set van der Waals radii according to Lewis dot structure |
| | 1 | set van der Waals radii according to Lewis dot structure **lewstr** (1st structure by default); see *GVB and Lewis Dot Structure Keywords* earlier in this section, and section 9.6 |
| **kesep** | 0 | combine terms for all one-electron matrices |
| | 1 | keep kinetic energy terms, nuclear attraction integrals, and point charge terms separate (note: if **isolv**=1 or 2, **kesep**=1 by default) |
| **isolvg** | 0 | compute gradients in solvation with method used for Jaguar version 3.5 and earlier |
| | ***1*** | compute gradients in solvation with more robust method for version 4.0 on |

are not indicated in bold italics, since the keywords' default values generally depend on other keywords. (By default, Jaguar calculations are performed in the gas phase, so **isolv**=0 and all other solvation keywords are irrelevant.)

The real-valued parameters for solvation calculations, which are shown in Table 8.6.17, help describe the solvent and the solute. Section 3.5 contains more details on these parameters. The default values for these parameters correspond to water.

***Table 8.6.17.*** *Real-valued solvation keywords.*

| keyword | default value | description |
|---|---|---|
| **epsout** | 80.37 | outer dielectric constant of solvent |
| **epsin** | 1.0 | inner dielectric constant of solvent [119] |

***Table 8.6.17.*** *(Continued) Real-valued solvation keywords.*

| keyword | default value | description |
|---|---|---|
| **radprb** | 1.40 | radius of solvent probe molecule |
| **sconv** | 0.00015 | solvation energy convergence criterion in Hartrees |
| **esolv0** | any number | gas phase energy of molecule, in Hartrees; used in some restart (new input) files for solvation jobs |

For solvated geometry optimizations, the **trust** keyword, which is described under *Geometry Optimization and Transition State Keywords* in this section, takes on a default value of 0.1 instead of its usual default of 0.3.

## *Properties Keywords*

Various keywords are used to request calculation of molecular properties, including multipole moments and charge fitting properties. Most of the keywords listed in Table 8.6.18 correspond to interface options described in Section 3.6. The default for each integer keyword is indicated in bold italics, and only the values listed in the table are allowed.

***Table 8.6.18.*** *Integer-valued keywords for charge fitting, multipole moment, and polarizability & hyperpolarizability calculations.*

| keyword | value | description |
|---|---|---|
| **icfit** | ***0*** | do not do electrostatic potential fitting |
| | 1 | fit electrostatic potential to atomic centers (default for solvation calculations) |
| | 2 | fit electrostatic potential to atomic centers and bond midpoints |
| **incdip** | ***0*** | use only total charge as constraint in electrostatic potential fitting |
| | 1 | use charge and dipole moment as constraints in electrostatic potential (ESP) fitting |
| | 11 | use charge, dipole moment, and quadrupole moment as constraints in electrostatic potential (ESP) fitting |

***Table 8.6.18.*** *(Continued) Integer-valued keywords for charge fitting, multipole moment, and polarizability & hyperpolarizability calculations.*

| keyword | value | description |
|---|---|---|
| | 111 | use charge, dipole moment, quadrupole moment, and octapole moment as constraints in electrostatic potential (ESP) fitting |
| | *ijk* | compute ESP fitted charges using total charge as a constraint, also constraining to dipole moment if $k=1$, to quadrupole moment if $j=1$, and to octapole moment if $i=1$ |
| | –1 | do all **incdip** options sequentially |
| **ldips** | ***1*** | do not calculate any multipole moments |
| | 2 | calculate dipole moments |
| | 3 | calculate dipole and quadrupole moments |
| | 4 | calculate dipole, quadrupole, and octapole moments |
| | 5 | calculate dipole, quadrupole, octapole, and hexadecapole moments |
| **ipolar** | ***0*** | do not calculate polarizabilities or hyperpolarizabilities |
| | –1 | calculate polarizabilities and hyperpolarizabilities using CPHF method |
| | 1 | calculate polarizabilities using 3-point finite field method |
| | 2 | calculate polarizabilities and hyperpolarizabilities using 3-point finite field method |
| | 5 | calculate polarizabilities and hyperpolarizabilities using 5-point finite field method |
| | 7 | calculate polarizabilities and hyperpolarizabilities using 7-point finite field method |

***Table 8.6.18.*** *(Continued) Integer-valued keywords for charge fitting, multipole moment, and polarizability & hyperpolarizability calculations.*

| keyword | value | description |
|---|---|---|
| **ldens** | ***0*** | do not calculate electron density |
| | 1 | calculate electron density on grid (grid choice set by grid keyword **geldens**; ultrafine grid used by default) |
| | –1 | calculate electron density on grid and write chdens file in a format that can be converted to a file Anthony Nicholls' program Grasp can read, using ps2grasp.f (available from Schrödinger; run with **geldens**=–3 and **denspc**=0.3 or smaller for best results) |
| **mulken** | ***0*** | do not calculate Mulliken populations |
| | 1 | calculate Mulliken populations by atom |
| | 2 | calculate Mulliken populations by basis function and by atom |
| | 3 | calculate Mulliken bond populations |

The finite field methods corresponding to **ipolar** > 0 differ in the data they use for numerical differentiation. The 3-point method uses the results from seven SCF calculations: one with no field, one with a field of E (whose input is described below) in the x direction, one with a field of –E in the x direction, and four others with fields of +E and –E, respectively, in the y and z directions, respectively. The 5-point method uses the same data as the 3-point method, as well as data from SCF calculations using fields of +aE and –aE, respectively, in the x, y, and z directions, respectively, where a is some constant. Similarly, the 7-point method uses the same data as the 3-point method, plus data obtained using fields of +aE, –aE, +bE, and –bE in the x, y, and z directions, where a and b are some constants. By default the magnitude of the electric field E is 0.024 au. If you want to use a different value, set the **efield** keyword to the desired value.

All polarizability methods are run with symmetry off—that is, the keyword **isymm** will be set to 0 automatically if **ipolar** ≠ 0. Similarly, for any polarizability calculation, the Methods keyword **econv**, which gives the energy convergence criterion, is set by default to $1.0 \times 10^{-6}$ (although if the calculation first satisfies the criterion dictated by the Methods keyword **dconv**, the energy convergence criterion is ignored).

The keywords **cfiterr**, **wispc**, **denspc**, and **efield**, which are listed in Table 8.6.19, take on real values. When charge fitting is constrained to reproduce multipole moments (that is, when **incdip**>0), the keyword **cfiterr** determines whether the multipole moment constraint is too restrictive to produce adequate charges: if the error in the total resultant charges is more than **cfiterr**, the charge fitting is rerun with a lower multipole moment constraint. The keyword **wispc** is used to set the spacing of the rectangular grid for ESP fitting when the grid keyword **gcharge**=–2. Similarly, the keyword **denspc** is used to set the spacing of the electron density rectangular grid when **ldens**=1 and the grid keyword **geldens**=–3. The **efield** keyword allows you to input an electric field for finite field polarizability and hyperpolarizability calculations, as described earlier in this subsection. Its default value shown in Table 8.6.19 applies to all cases when **ipolar** > 1. For **ipolar**=1 (3-point, polarizability-only calculations), **efield**'s default value is 0.006 A.U.

***Table 8.6.19.*** *Real-valued property keywords.*

| keyword | default value | description |
|---|---|---|
| **cfiterr** | $1.0 \times 10^{-6}$ | allowed error in ESP charge fitting when fitting is constrained to reproduce multipole moments |
| **wispc** | 0.75 | spacing in bohr of rectangular grid for ESP fitting |
| **denspc** | 0.75 | spacing in bohr of rectangular grid for electron density calculation |
| **efield** | 0.024 | electric field for polarizability and hyperpolarizability calculations, in A.U. (default is 0.006 for **ipolar**=1) |

If you want to print out the electrostatic potential at gridpoints which you specify, add the keyword settings **gcharge**=-6 and **ip172**=2 to the **gen** section of your input file. The **gcharge**=-6 setting causes Jaguar to use the gridpoints and weighting factors in a file whose name and location are specified by the GPTSFILE line in the input file (see Section 8.2). The **ip172**=2 setting causes Jaguar to write out a file with the name <jobname>.resp containing the ESP data (see the text under Table 8.6.27).

## *Frequency-Related Keywords*

For jobs that include a calculation of vibrational frequencies, various frequency-related properties can also be computed by setting the

appropriate keywords. Most of these keywords, which are listed in Table 8.6.20, correspond to interface options described in Section 3.7. Keyword values in bold italics are the default values, and only the values listed in the table are allowed.

***Table 8.6.20.*** *Keywords for frequency-related properties.*

| keyword | value | description |
|---|---|---|
| **ifreq** | ***0*** | do not calculate frequencies (second derivatives) |
| | 1 | get frequencies from Hessian of second derivatives of energy |
| | –1 | calculate frequencies from most recent Hessian (from end of optimization or from initial Hessian if initial Hessian was never updated) |
| **imw** | ***0*** | print normal modes in cartesian coordinates without mass-weighting |
| | 1 | print normal modes in mass-weighted cartesian coordinates |
| **isqm** | 0 | do not scale frequencies using Pulay's Modified Scaled Quantum Mechanical Force Fields (SQM) method |
| | 1 | scale frequencies using Pulay's SQM method, and use scaled frequencies for thermochemical calculations (only allowed for B3LYP calculations with the 6-31G* basis set) |
| **scalfr** | >0 | scale vibrational frequencies by this factor (default is 1.0), and use scaled frequencies for thermochemical calculations |
| **irder** | ***0*** | do not compute dipole derivatives or IR intensities for vibrational frequencies |
| | 1 | compute derivatives of dipole moment and IR intensities for vibrational frequencies (see text for details) |
| **press** | >0 | pressure for thermochemical calculations from frequencies, in atm; default is 1.0 |
| **tmpini** | >0 | initial temperature for thermochemical calculations, in K; default is 298.15 |

***Table 8.6.20.*** *(Continued) Keywords for frequency-related properties.*

| keyword | value | description |
|---|---|---|
| **tmpstp** | >0 | temperature step size (difference between consecutive temperatures) for thermochemical calculations, in K; default is 10.0 |
| **ntemp** | >0 | number of temperatures at which thermochemical properties are computed; default is 1 |

The thermochemical properties are listed in cal/mol K and kcal/mol by default. Use the output option **ip28**=2 for output in J/mol K and kJ/mol.

When the calculation of vibrational frequencies is requested with **ifreq**=1 and the level of theory being used is Hartree-Fock, IR intensities for the IR-active vibrational modes will be automatically calculated (i.e., **irder**=1 automatically). For DFT, you must explicitly set **irder**=1, and the derivatives must be calculated numerically by setting **nmder**=2. The calculation of IR intensities involves the calculation of the dipole moment derivatives. If you only want to calculate dipole moment derivatives but don't want to do the expensive frequency calculation that is normally required to get them, you must set up a special **path** section (see Section 8.17) to tell Jaguar the appropriate sequence of executables to run in order to calculate dipole derivatives only. The **path** section to use is:

&path pre onee hfig probe grid rwr scf ira rwr irb &

You must also set **irder**=1 and **isymm**=0 and **igeopt**=1. The **igeopt** setting is necessary to force tight accuracy in the SCF, but no optimization will actually be performed.

## *Basis Set Keywords*

The character string keyword **basis** allows you to override the default basis set (6-31G**). This keyword should be a string describing the standard basis and any desired polarization and diffuse functions. The string describing the standard basis should be chosen from the first column of Table 3.8.1 or Table 3.8.2 in Section 3.8. Lowercase or uppercase letters can be used. The polarization and diffuse function options are described by adding *, **, +, or ++ immediately after the basis name. The meaning of these symbols is also described in Section 3.8. Neither polarization nor diffuse functions are used if none of these options are specified. The tables in Section 3.8 list the

basis sets and indicate which options and atoms Jaguar currently accepts for each.

The other keyword relating to the basis set, **numd**, allows you to choose whether to use five or six d functions in each d shell. If you do not set **numd** explicitly, the number of d functions is set automatically, depending on the basis set, as described in Section 3.8. Possible settings for **numd** are shown in Table 8.6.21.

***Table 8.6.21.*** *Keyword to determine the number of d functions.*

| keyword | value | description |
|---|---|---|
| **numd** | 5 | use 5 d functions, regardless of basis set |
| | 6 | use 6 d functions, regardless of basis set |

## *Keywords for SCF Methods*

Many of the keywords which control the SCF calculation can be set from the Methods window in the interface, as described in Section 3.9. (The other keyword settings corresponding to Methods window settings are described under *Initial Guess Keywords* and *Localization Keywords* in this section.)

The two real-valued convergence criterion keywords are **econv**, the energy convergence criterion, which dictates the maximum difference in energy between one SCF iteration and the next for convergence to be assumed, and **dconv**, the criterion for the root mean square of the change in density matrix elements. The default value of **econv** is normally $5.0 \times 10^{-5}$ Hartrees. However, for polarizability or hyperpolarizability calculations, **econv** is $1.0 \times 10^{-6}$ Hartrees by default. When the root mean squared change in density matrix elements for a polarizability, hyperpolarizability, or geometry optimization calculation is less than **dconv**, whose default value is $5.0 \times 10^{-6}$, the calculation is considered to have converged.

By default, calculations use a DIIS (or GVB-DIIS) convergence scheme, meaning that every iteration, an estimate of the Fock matrix is produced which is a linear combination of previous Fock matrices, including the one calculated during that iteration. The keyword **maxdiis**, which has a value of 10 by default, indicates the maximum number of previous Fock matrices that will be used for this scheme during any iteration. The keyword **stdiis** gives an error criterion; DIIS is started when the largest value of the DIIS error vector is less than the value of **stdiis**, whose units are hartrees. For standard Hartree-Fock, DFT, LMP2, or GVB-LMP2 calculations, **stdiis** is 2.0 by

default; for GVB calculations (when **iconv**, listed in Table 8.6.22, is 3 or 4), **stdiis** is 1.0 by default; and for cases involving transition metals or open shell calculations, or when the initial guess is obtained from the one-electron Hamiltonian (**iguess**=0; see the subsection *Initial Guess Keywords*), its default value is 0.1. In general, after GVB-DIIS starts, any density matrix averaging requested by the keywords **iteravg** and **istavg** (explained in Table 8.6.22) will be turned off.

The last real-valued methods keyword, **vshift**, describes the amount the virtual orbitals' energies are increased before diagonalization, in atomic units. This keyword can be used to reduce mixing of the real and virtual orbitals, which sometimes helps convergence. By default, **vshift** is zero, except for DFT calculations, when the default is 0.2 (for hybrid methods) or 0.3 (for non-hybrid methods). Non-default values should probably be on the order of 0.1–0.5.

The other integer SCF keywords are described in Table 8.6.22. As usual, the default values are in bold italics. Note, however, that the

***Table 8.6.22.*** *Integer keywords relating to methods used in the SCF convergence procedures.*

| keyword | value | description |
|---|---|---|
| **iuhf** | ***0*** | restricted open-shell (ROHF or RODFT) calculation |
| | 1 | unrestricted (UHF or UDFT) calculation |
| **iconv** | 0 | convergence via Fock matrix diagonalization |
| | ***1*** | DIIS convergence scheme (default choice for most non-GVB calculations; see **iconv**=4) |
| | 3 | GVB2P5-style convergence (OCBSE) |
| | 4 | GVB-DIIS convergence (default choice for GVB, open shell singlet calculations, and calculations whose initial guess is obtained from $\boldsymbol{H}_0$) |
| **maxit** | 0 | calculate energy, but do not update wavefunction (i.e. do only one iteration) |
| | >0 | perform a maximum of maxit SCF iterations (default value is 48) |

***Table 8.6.22.*** *(Continued) Integer keywords relating to methods used in the SCF convergence procedures.*

| keyword | value | description |
|---|---|---|
| **newcon** | 0 | use physical constants and conversion factors equivalent to those used in GAUSSIAN 86 |
| | ***1*** | use physical constants and conversion factors equivalent to those used in GAUSSIAN 88, 90, 92 |
| | –1 | use GVB2P5 physical constants and conversion factors |
| **iacc** | 1 | only ultrafine grid used; "tight" cutoffs |
| | 2 | accurate: mixed grid types, accurate cutoffs (default choice for transition metals, sometimes for other atoms beyond Ar) |
| | ***3*** | quick: mixed grid types, looser cutoffs |
| **iacscf** | ***0*** | make no special adjustments (variable vshift, cutoff adjustments, etc.) for convergence |
| | 1 | start with level shift (**vshift)** of 5.0 and decrement over 10 iterations to 0.8; keep number of canonical orbitals fixed during optimization; run at ultrafine accuracy level and with tight cutoffs |
| | 2 | start with level shift (**vshift**) of 6.0 and decrement over 12 iterations to 0.8; vary level shift during run by raising it when SCF is restarted (here, when the energy rises by 0.0001 a.u.) |
| | 3 | use extreme cutoffs (maximal analytic corrections) while still allowing medium pseudospectral grids for some iterations |
| | 4 | same as **iacscf=**1, except with maximal analytic corrections |
| **jksep** | ***0*** | 2***J*** – ***K*** formed for core when convenient |
| | 1 | ***J*** and ***K*** for core are kept separate |
| **noatcor** | ***0*** | analytic corrections calculated |
| | 1 | no analytic corrections calculated |

***Table 8.6.22.*** *(Continued) Integer keywords relating to methods used in the SCF convergence procedures.*

| keyword | value | description |
|---|---|---|
| **nops** | ***0*** | use pseudospectral method to calculate ***J*** and ***K*** operators |
| | 1 | construct ***J*** and ***K*** from analytic four-center two-electron integrals (no grid used |
| **noupdat** | ***0*** | Fock matrix updating [108] set on or off automatically |
| | 1 | no Fock matrix updating (set **iacc**=1 if you set noupdat=1) |
| **iteravg** | ***0*** | do not average density matrices and adjust orbitals accordingly (unless **istavg** keyword requests averaging) |
| | >0 | for iterations whose number is n***iteravg** + 1, where n is an integer, revise orbitals so that they correspond to average of density matrices from preceding and current iterations |
| **istavg** | ***0*** | do not average density matrices and adjust orbitals accordingly (unless **iteravg** keyword requests averaging) |
| | >0 | for iterations whose number is **istavg**, revise orbitals so that they correspond to average of density matrices from preceding and current iterations |
| **noauto** | ***0*** | grid choice is automatic |
| | 1 | all calculations done on coarse grid |
| | 2 | all calculations done on medium grid |
| | 3 | all calculations done on fine grid |
| | 4 | all calculations done on ultrafine grid |
| **idenavg** | ***0*** | converge in the usual fashion |
| | 1 | do density averaging before DIIS starts, mixing in some of old orbitals with new orbitals (default for DFT calculations) |
| **lastwv** | 0 | skip diagonalization of Fock matrix on last iteration |
| | ***1*** | diagonalize Fock matrix on last iteration |

***Table 8.6.22.*** *(Continued) Integer keywords relating to methods used in the SCF convergence procedures.*

| keyword | value | description |
|---|---|---|
| **nosuper** | ***0*** | evaluate integrals simultaneously over s and p basis functions with the same exponents ("superblocks") |
| | 1 | evaluate integrals separately for s and p basis functions which have the same exponents |
| | 2 | use superblocks for all integrals except for gradient |
| **itwice** | ***1*** | do $A_{mng}$ integrals once in SCF routine |
| | 2 | do $A_{mng}$ integrals twice in SCF routine—required for GVB, optional for HF |
| **ichange** | ≠0 | change all cutoffs (except those related to ***S*** eigenvalues, bc pairs, or ab distance cutoff for exchange) by a factor of 10 to the **ichange** power |

keyword settings for convergence are somewhat complicated, and the defaults vary somewhat depending upon the settings of other keywords.

One of the most important keywords in controlling the SCF is **iacscf**. This keyword should be employed when the SCF fails to converge under the default conditions, especially for transition metal-containing systems or clusters. Start with **iacscf** = 1 and if that does not work then try **iacscf** = 4. **iacscf** = 2 was developed especially for hemes and related molecules, while **iacscf** = 3 was effective for graphitic systems. Energies obtained with **iacscf** = 2 can be directly compared to energies obtained without using **iacscf**. Energies obtained using other values of **iacscf** are not comparable because they use different grids or cutoffs.

## *Initial Guess Keywords*

Table 8.6.23 lists the keywords related to the initial guess and the meaning of the integer values each keyword can take on. Most of the keyword values in Table 8.6.23 correspond to interface options described in Section 3.9. Default choices are indicated in bold italics.

***Table 8.6.23.*** *Initial guess keywords.*

| keyword | value | description |
|---|---|---|
| **igonly** | ***0*** | no effect |
| | 1 | use initial guess or input wavefunction for any post-SCF calculations, skipping SCF step |
| **iguess** | 0 | generate initial guess by diagonalizing one-electron Hamiltonian |
| | 1 | read initial guess from **guess** section from input file or from guess file specified in WAVEFNFILE line (**iguess**=1 automatically if input file contains non-empty **guess** section) |
| | 3 | read initial guess from GVB2P5 .trn file |
| | ***10*** | construct initial guess from orbitals which give best overlap with atomic orbitals in default.atomig (or other .atomig file listed in input file) obtained by SCF calculations on atoms (note that if **guess** section exists, this is not the default choice) |
| | 11 | construct initial guess from orbitals whose densities, when summed, best agree with the sum of the densities of the atomic orbitals in default.atomig (or other atomig file listed in input file) obtained by SCF calculations on atoms |
| | 25 | for a transition-metal-containing system, construct a high-quality initial guess using ligand field theory as described in reference [19] |
| | 30 | for a transition-metal-containing system, construct a high-quality initial guess using ligand field theory (including d-d repulsion), as described in reference [19] |
| **ihfgvb** | ***0*** | a) read in GVB initial guess from **guess** section if **iguess**=1, and do not run hfig or gvbig programs, or b) compute it from HF initial guess (whose origin is determined by **iguess**) if **iguess** is not 1 |

***Table 8.6.23.*** *(Continued) Initial guess keywords.*

| keyword | value | description |
|---|---|---|
| | 1 | converge HF wavefunction (where the initial guess is determined by **iguess**) and use converged HF wavefunction as input to program gvbig to get GVB initial guess |
| | 2 | calculate a GVB initial guess from HF initial guess, whose origin is determined by setting **iguess** |
| **ihamtyp** | ***0*** | construct Hamiltonian using standard core, open, and GVB orbitals |
| | 2 | make highest two orbitals in initial guess an open shell singlet pair (ROHF only; for UHF and UDFT, use **iopt457**=457, **isymm**=0 and **iuhf**=1) |
| | 3 | input Hamiltonian in **ham** section (**ihamtyp**=3 by default if a non-empty **ham** section exists) |

## *Localization Keywords*

For any Jaguar job, the final wavefunction can be localized after it is computed. (Localization can also be used to provide localized orbitals for the LMP2 method; see *LMP2 Keywords* in this section for details.) The keywords in Table 8.6.24 describe the available options for final wavefunction localization. See *Final Localization of the Orbitals* in Section 3.9 for a description of the localization methods and the interface settings related to localization.

***Table 8.6.24.*** *Keywords related to localization of orbitals.*

| keyword | value | description |
|---|---|---|
| **locpostc** | ***0*** | do not localize core orbitals of final wavefunction |
| | 1 | perform Boys localization on core orbitals of final wavefunction |
| | 2 | perform Pipek-Mezey localization on core orbitals of final wavefunction, maximizing Mulliken atomic populations |

***Table 8.6.24.*** *Keywords related to localization of orbitals.*

| keyword | value | description |
|---|---|---|
| | 3 | perform Pipek-Mezey localization on core orbitals of final wavefunction, maximizing Mulliken basis function populations |
| | –1 | mix the core and valence orbitals before localization, then localize according to the **locpostv** setting |
| **locpostv** | ***0*** | do not localize valence orbitals of final wavefunction |
| | 1 | perform Boys localization on valence orbitals of final wavefunction |
| | 2 | perform Pipek-Mezey localization on valence orbitals of final wavefunction, maximizing Mulliken atomic populations |
| | 3 | perform Pipek-Mezey localization on valence orbitals of final wavefunction, maximizing Mulliken basis function populations |
| **iordboy** | 0 | do not order orbitals at end of Boys localization |
| | 1 | order orbitals by their one-electron energy at the end of Boys localization |
| **ixtrboy** | ***0*** | do not try to diagonalize multiple-bond orbitals at the end of the Boys localization |
| | 1 | try to diagonalize multiple-bond orbitals at the end of the Boys localization—see text in this subsection |

When the keyword **ixtrboy** described in Table 8.6.24 is set to 1, an additional procedure is added on to the Boys localization process. Because the Boys orbitals may be unphysical for multiple bonds, since they create multiple "banana" bonds between pairs of atoms rather than forming sigma-like, pi-like, and related orbitals, the Boys orbitals for multiple bonds may be diagonalized using the one-electron Hamiltonians. The output for this procedure begins with a table of the Mulliken populations for each orbital on each atom, which reveals multiple bonds, as described in the following table. Every "bond pair space" made up of all orbitals with significant Mulliken populations on the same pair of atoms is diagonalized, and the output indicates the number of these bond pair spaces found and the ordering

of the new orbitals by their one-electron Hamiltonian values. If you choose to print out Boys orbitals by setting the print keyword **ip107** to 2, it is these final orbitals which are printed.

## *Using Babel to Output Files*

You can call the program Babel [24] from Jaguar to generate output in any of a variety of formats, although the files produced by Babel will contain only geometries, omitting any calculation settings. The output can be generated at the end of each iteration in a geometry optimization or at the end of any job. To generate such an output file, you must first know the extension that corresponds to the appropriate file type (for instance, "spar" for SPARTAN). The extensions and file types supported are shown in Table 8.6.25.

***Table 8.6.25.*** *Keyword settings for the* ***babel*** *and* ***babelg*** *keywords, and the corresponding output file types.*

| output extension | corresponding file type |
|---|---|
| t | Alchemy |
| bs | Ball and Stick |
| caccrt | Cacao Cartesian |
| cache | CAChe MolStruct |
| c3d1 | Chem3D Cartesian 1 |
| c3d2 | Chem3D Cartesian 2 |
| d | ChemDraw Conn. Table |
| cssr | CSD CSSR |
| diag | DIAGNOTICS |
| gamin | GAMESS Input |
| gcart | GAUSSIAN Cartesian |
| g | GAUSSIAN Z-matrix |
| hin | Hyperchem HIN |
| i | IDATM |
| macmol | Mac Molecule |
| k | Macromodel |
| micro | Micro World |
| mi | MM2 Input |
| mo | MM2 Ouput |

***Table 8.6.25.*** *(Continued) Keyword settings for the* ***babel*** *and* ***babelg*** *keywords, and the corresponding output file types.*

| output extension | corresponding file type |
|---|---|
| mm3 | MM3 |
| mmads | MMADS |
| mdl | MDL Mol |
| ac | MOPAC Cartesian |
| ai | MOPAC Internal |
| p | PDB |
| psz | Jaguar Z-matrix file |
| psc | Jaguar Cartesian file |
| report | Report |
| spar | SPARTAN |
| mol | Sybyl Mol |
| mol2 | Sybyl Mol2 |
| x | XYZ |

If you want to generate an output file in a particular format only at the end of a job, you should use a keyword setting of the form **babel**=*outext*, where *outext* is one of the possible output extensions listed in Table 8.6.25. You can set **babel** more than once (using separate **babel**=*outext* assignments) if you want to generate several files.

To generate output files at the end of all iterations in a minimum-energy structure or transition state optimization as the optimization proceeds, set the **babelg** keyword to the appropriate output extension string or strings. Like the **babel** keyword, the **babelg** keyword can be set more than once to generate files in several formats.

As files are generated with Babel during Jaguar runs, they are immediately copied back to the relevant output directory. Files generated from jobs with **babel** keyword settings have names of the form *jobname.outext* (for instance, "h2o.spar"), where *jobname* is the usual job name and *outext* is an output extension, as described above; files generated from geometry optimizations with **babelg** settings have names of the form *jobname#.outext*, where # is a four-digit number corresponding to the iteration number (for example, 0001 for the first geometry iteration), and all letters in the job name are converted to lower case by Babel. Note that you can use a **babelg** keyword setting

to output structures generated during an optimization as the optimization proceeds.

For either **babel** or **babelg** keyword settings, you can use an optional extra extension for the file name by setting **babel** (or **babelg**) to a keyword in the form *outext.opt,* where *opt* is any extension you wish to use. For instance, if you made the setting **babel**=g.g92 in a Jaguar input file called h2o.in, the resulting job would create a Gaussian input file called h2o.g.g92.

## *Standard Output Keywords*

The keywords listed in Table 8.6.26 are the standard printout options. They are all set to 1 by default, and the result is that none of the information that the keywords access is output. Many of the printout options can be turned on from the interface, as described in section 5.4.

The keyword setting **ip6**=3 provides much more detailed timing information that the setting **ip6**=2. Similarly, the keyword setting **ip192**=3 provides more detailed output than **ip192**=2; the **ip192**=3 setting includes the hessian.

The keyword setting **kesep**=1, which is normally part of a solvation calculation (see Table 8.6.16), will cause the virial ratio, -V/T, to be printed out at the end of each SCF.

***Table 8.6.26.*** *Ouput keywords and their settings. When any of the following keywords is set equal to 1, the corresponding output is not generated.*

| keyword | value | description |
|---|---|---|
| **ip1** | 2 | Gaussian function list for basis set |
| **ip3** | 2 | Gaussian function list for dealiasing functions |
| **ip4** | 2 | number of dealiasing functions used |
| **ip5** | 2 | memory, disk, and i/o information |
| **ip6** | 2 | timing information (cpu and i/o) |
| | 3 | more detailed timing information than when **ip6** is set to 2 |
| | -2 | timing information (cpu and wall clock) |
| **ip7** | 2 | grid shell locations |
| **ip8** | 2 | Gaussian function list for derivatives of basis functions |

***Table 8.6.26.*** *(Continued) Ouput keywords and their settings. When any of the following keywords is set equal to 1, the corresponding output is not generated.*

| keyword | value | description |
|---|---|---|
| **ip11** | 2 | bond lengths and angles |
| | 3 | same as setting **ip11**=2, but includes all internuclear distances (regardless of connectivity) and torsions |
| | 4 | same as setting **ip11**=3, but includes all possible angles (regardless of atom connectivity) |
| | 5 | same as setting **ip11**=4, but includes all possible torsions (regardless of atom connectivity) |
| **ip12** | 2 | connectivity table |
| **ip13** | 2 | eigenvectors and eigenvalues of overlap matrix |
| **ip18** | 2 | overlap matrix |
| **ip19** | 2 | one-electron Hamiltonian |
| **ip24** | 2 | *all* keyword settings, including internal ones |
| **ip25** | 2 | multipole moments in atomic units (and Debye) |
| **ip26** | 2 | geometries in bohr as well as Angstroms |
| **ip70** | 2 | extra geometry optimization details |
| **ip170** | 2 | localized orbital locations and LMP2 pair energies for local LMP2 calculations (full local LMP2 energy correction is sum of pair energies) |
| **ip173** | 2 | Fock matrix in Boys-localized orbital space |
| **ip192** | 2 | extra optimization-related information, such as the quadratic energy error |
| | 3 | same as setting **ip192**=2, but includes more detailed information such as the Hessian |
| **ip193** | 2 | numerical Hessian in freq output |
| **ip194** | 2 | diagonal force constants in internal coordinates |

***Table 8.6.26.*** *(Continued) Ouput keywords and their settings. When any of the following keywords is set equal to 1, the corresponding output is not generated.*

| keyword | value | description |
|---|---|---|
| | 3 | same as setting **ip194=**2, but also includes off-diagonal force constants if they are larger than a factor (0.01 by default) times the geometric mean of the corresponding off-diagonal elements; the value of the factor can be set using the **opt194** keyword |
| | 4 | all diagonal and off-diagonal force constants are printed |

## *Files Output Keywords*

The files output keywords are the printout options which cause files other than the usual log and output files to be created. All but one of these keywords are set to 1 by default, generally indicating that ordinarily none of this information is output.

The files output keyword **ip151** controls whether or not a Jaguar restart file is written. It is the only files output keyword whose default value of 1 indicates that it is on. When **ip151** is set to 1, the file restart.in is created in the temp directory for the job at the end of the last completed Jaguar program, writing over any previously generated restart.in file for the job. The restart.in file contains the results from the run, including the new geometry if the run that produced it was a geometry optimization. This input file can therefore be used to restart the calculation. At the end of the job, the restart input file is copied to your local job directory (under the name <jobname>.01.in, unless that file already exists, otherwise <jobname>.02.in or <jobname>.03.in, and so on). To turn off **ip151**, you must set it to 0.

The other files output keywords control whether files for various other programs such as GAMESS are written out during a Jaguar job. The effect of setting each of these keywords to 2 is shown in Table 8.6.27. Many of these printout options can be turned on from the interface, as described in section 5.5.

Additional settings are available for **ip160** and **ip165**. When **ip165** is 3, the SPARTAN 4.0 archive file is written to the local job directory as *jobname*.arc. When **ip160** equals 3, an initial guess is included in the .g92 file generated by the run (by default, .g92 files generated for GVB calculations include initial guesses, but those for other calculations do not). If SCF iterations are performed, the initial guess for the

***Table 8.6.27.*** *Effect of setting output keywords for files to 2. See text in this subsection for information on* ***ip151*** *and information on other options for* ***ip160****.*

| keyword | description of what is printed when ip*i* = 2 |
|---|---|
| **ip90** | Molden orbitals file (.molf file) |
| **ip160** | GAUSSIAN input file (.g92 file) (see text for **ip160**=3, 4, or 5) |
| **ip161** | GVB2P5 trans matrix (.trn file) |
| **ip163** | GAUSSIAN basis set (.gbs file) |
| **ip164** | MQM basis set (.bas file) |
| **ip165** | SPARTAN 4.0 archive file; appears in temp directory as spart.arc (to write .arc file to local job directory instead, use **ip165**=3) |
| **ip168** | GAMESS input file (.gamess file) |
| **ip172** | RESP (Restrained Electrostatic Potential [109]) file (.resp file) (set to 3 to include grid weights) |
| **ip175** | XMol file (.xyz file) with geometries generated during optimization |
| **ip177** | AIMPAC (.wfn file) which works with RHF/ ROHF but not UHF |

.g92 file will be obtained from the resulting wavefunction; otherwise, it will be generated from the appropriate Jaguar initial guess routine. When **ip160** equals 5, the basis set is included explicitly within the .g92 file, rather than just the basis set name. When it equals 4, the trial wavefunction and the basis set are included.

The format of the .resp file created with the **ip172** keyword is as follows. The first line contains the number atoms in the molecule and the number of gridpoints at which the ESP was evaluated, respectively. Then the cartesian coordinates of the atoms, in bohrs, are given. Each of the remaining lines contains the ESP (in hartrees), the coordinates of the gridpoint (in bohrs) at which the ESP was evaluated, and, if **ip172**=3, the grid weights.

## *Output Keywords for Each Iteration*

The information in Table 8.6.28 concerns output which can be printed out every SCF iteration if the keyword is set to 2. section 5.6 describes how to turn on these settings from the interface. Once again, setting these keywords to 1 generally indicates that the information will not be printed.

***Table 8.6.28.*** *Effect of setting output keywords for each iteration to 2. See text in this subsection for information on* ***ip152****.*

| keyword | description of what is printed when ip*i* = 2 |
|---|---|
| **ip15** | DIIS coefficients |
| **ip17** | energy components |
| **ip110** | density matrix (if Fock matrix updating was not performed during that iteration) or differential density matrix (if Fock matrix updating was done) |
| **ip121** | all ***J*** and ***K*** matrices, in atomic orbital space |
| **ip122** | Fock matrix in atomic orbital space (HF) or molecular orbital space (GVB) |
| **ip123** | Fock matrix in canonical orbital space |
| **ip149** | GVB data: f, a, b, etc. |
| **ip188** | debug printing for automatic cutoff/ convergence scheme |
| **ip201** | total electronic density integrated on the DFT grid |

The option **ip152** is the only one whose default value of 1 indicates that it is on. When **ip152** is set to 1, the file restart.in is created in the temp directory for the job at the end of the last completed iteration (overwriting the restart.in file created from the previous iteration). This input file can then be used to restart the calculation. To turn off **ip152**, you must set it to 0.

## *Orbital Output Keywords*

Orbital information can be printed out as well. The orbital keywords determine what orbitals are printed in the output, at what stage they are printed, and the format in which the orbital output appears.

The keyword **ipvirt** determines how many of the virtual orbitals are printed in the output file and in the restart (new input) file. Virtual orbitals are printed in order of increasing energy. The virtual orbitals are obtained by diagonalizing $H_0 + \sum f(2J - K)$, where $f$ is the occupation of each orbital (1 for a closed shell). If **ipvirt**=–1, all virtual orbitals are printed in the output and restart files; otherwise, **ipvirt** virtual orbitals are printed (if that many virtual orbitals exist). By default, **ipvirt**=10.

Several possible formats and levels of information can be requested for each other keyword determining the orbitals printed. The choice of keywords, which are listed in Table 8.6.29, determines the stage (or stages) at which orbitals are printed; the keyword values determine

***Table 8.6.29.*** *Keywords to specify when to output orbitals.*

| keyword | prints orbitals: |
|---|---|
| **ip100** | for initial guess from before SCF (generally redundant with **ip105**) |
| **ip101** | in canonical orbital space (each SCF iteration) |
| **ip102** | at end of job |
| **ip103** | in atomic orbital space (each SCF iteration) |
| **ip104** | in atomic orbital space after SCF |
| **ip105** | for HF initial guess |
| **ip106** | for GVB initial guess |
| **ip107** | after Boys or Pipek localization |

which orbitals are printed and the format of the printing. These settings can generally also be made from the interface, as described in section 5.7.

Table 8.6.30 explains the possible values for the orbital output options, aside from 1, the default, which turns off printing. The variable **n** in the table can be either 0, 5, or 10. If it is 0, all occupied orbitals, including GVB natural orbitals, are printed. If **n** is 5, all occupied orbitals and **ipvirt** virtual orbitals are printed (or all virtuals if **ipvirt**=–1). Setting **n** to 10 causes only the GVB non-orthogonal orbitals to be printed.

For example, 'ip106=10' would mean that all orbitals were to be printed in FORTRAN f8.5 format after the GVB initial guess was created. The options **ip105** ≥ 12 are not valid; use **ip100** instead. In canonical orbital space, the atom and function type labels are meaningless. If a keyword is set to 4, 5, 9, or 10, the results are suitable for input in the **guess** section or for input to GAUSSIAN (guess=cards) or GVB2P5.

When the orbital output is in table form, each function's coefficient for each orbital is shown, with the functions shown in numbered rows and the orbitals in numbered columns. When it is in list form, each orbital is listed in turn, with the function coefficients listed in order. When **ip*x*** = 2 + **n**, only coefficients larger than a particular value

***Table 8.6.30.*** *Format and type of orbital output. The value of* ***n*** *determines which orbitals (e.g., occupied) are printed;* ***x*** *determines the stage at which orbitals are printed (see Table 8.6.29).*

| | **ip*x* = 2 + n** | **ip*x* = 3 + n** | **ip*x* = 4 + n** | **ip*x* = 5 + n** | **ip*x* = 6 + n** |
|---|---|---|---|---|---|
| **format** | f5.2 | f10.5 | f19.15 | f8.5 | e15.6 |
| **atom, basis function type shown** | Y | Y | N | N | N |
| **orbital occupation indicated** | Y | N | Y | Y | N |
| **coefficients printed** | large | all | all | all | all |
| **form shown** | list | table | list | list | table |

(generally .05) are listed, and the atom identifiers (for instance, h2) and function types (for instance, S for *s*, Z for $p_z$, or XX for $d_{xx}$) are shown. When **ip*x*** = 4 + **n** or **ip*x*** = 5 + **n**, all coefficients are listed, in order but without numbering.

For examples of the output that shows up in the output file for a calculation of water with a 6-31G** basis set for various values of **ip104**, see the five examples given at the end of section 5.7. The five examples correspond to **ip104**=2, **ip104**=3, **ip104**=4, **ip104**=5, and **ip104**=6, in that order. Only the first two occupied orbitals are shown in each case, and not all functions are shown; those gaps are indicated by [...].

## *Grid and Dealiasing Function Keywords*

The grid and dealiasing function keywords allow the user to select from among the various sets of grids and dealiasing functions available in the grid and dealiasing (.grid and .daf) input files, which are described in section 9.3 and section 9.4, and from the grids generated within Jaguar. These keywords are used to specify which grid or dealiasing sets correspond to particular descriptions; this correspondence is often indicated by keyword values depending on the order of sets in the grid and dealiasing input files.

For density functional theory calculations, the grid keywords **gdftmed**, **gdftfine**, and **gdftgrad** determine the point density for the DFT grids used for the SCF and gradient calculations, respectively. If these keywords are set to –10, –11, and –12, respectively, the standard, default grid is used. A value of –13 corresponds to a finer grid.

Table 8.6.31 shows the types of grids that can be specified for

***Table 8.6.31.*** *Grid (or dealiasing function) types for Jaguar (for non-DFT portions of calculations). These names are used in the grid-related keywords described in Table 8.6.32.*

| *name* | description |
|---|---|
| **coarse** | least expensive, least accurate level |
| **medium** | used for most SCF iterations |
| **fine** | sometimes used for a limited number of iterations |
| **ufine** | ultrafine; most accurate level |
| **grad** | used in gradient computation |
| **mp2** | grid used for LMP2 portion of LMP2 calculations |
| **charge** | grid used for charge fitting |
| **eldens** | used for electron density calculations |
| **cphf** | grid used for coupled perturbed HF (CPHF) calculations |

portions of the calculation that do not involve density functional theory. Generally, these grid types are used for pseudospectral SCF iterations or for charge fitting.

The grid-related keywords and their allowed and default values are described in Table 8.6.32, where ***name*** corresponds to one of the grid types listed in Table 8.6.31. As an example, 'gmedium=2' indicates

***Table 8.6.32.*** *Keywords for specification of length scales for sorting of basis functions, grid usage, and dealiasing function usage.*

| keyword | value | description | default for: |
|---|---|---|---|
| **l*name*** | 1 | only one length scale used for calculation | **lcoarse** |
| | 2 | basis functions are sorted into short- and long-range | **lmedium, lfine, lufine, lgrad** |
| ***g*name** | >0 | specifies which parameter set from .grid file should be used for grid (e.g. 2 for second) | **gcoarse** (1), **gmedium** (2), **gfine** (3), **gufine** (4), **ggrad** (4), **gmp2** (4), **geldens** (4), **gcphf** (2) |

***Table 8.6.32.*** *(Continued) Keywords for specification of length scales for sorting of basis functions, grid usage, and dealiasing function usage.*

| keyword | value | description | default for: |
|---|---|---|---|
| | –1 | use spherical charge fitting grid generated within Jaguar for grid listed by ***name*** | **gcharge** |
| | –2 | use cubic charge fitting grid generated within Jaguar for grid listed by ***name*** | none |
| | –3 | use cubic electron density grid generated within Jaguar for grid listed by ***name*** | none |
| | –6 | use grid and weights from file specified by GPTSFILE line in input file for grid listed by ***name*** | none |
| **d*name*** | >0 | specifies which dealiasing function from .daf file should be used | **dcoarse** (1), **dmedium** (2), **dfine** (3), **dufine** (4), **dgrad** (5) |

that the medium grid to be used for the calculation is the second one listed in the .grid file, while 'geldens=–3' indicates that an electron density calculation should use a cubic grid.

You can input your own set of grid points and weights by using the **g*name*=**–6 option and the GPTSFILE line of the input file, which is described in Section 8.2.

## *Memory Usage Keywords*

Some of the memory usage for the program can be controlled through keywords. These keywords may be particularly useful if you are experiencing problems running jobs due to memory-related failures, as described in the troubleshooting information in section 10.3.

Memory usage keywords are listed in Table 8.6.33, along with their default values and a description of their uses. Users who want to change some memory usage but who do not have a detailed knowl-

***Table 8.6.33.*** *Keywords related to memory and disk usage.*

| keyword | default | description |
|---|---|---|
| **mxstrip** | 200 | Information for matrix elements evaluated on basis functions stored in core in strips of **mxstrip***N words, rather than $N^2$ words at a time (where N is the number of basis functions). |
| **mxpage** | 1000 | For pseudospectral evaluation of ***J*** and ***K*** on grid points in program scf, memory is allocated ngblok***mxpage** words at a time as needed, where ngblok is a parameter currently set to 640 on Crays & 128 on other machines. |
| **nbuck** | 64 (256 on Crays) | Gridblocks are split up into sub-gridblocks whose points are all on the same atom and in the same region of space, with at most **nbuck** points, where **nbuck** ≤ ngblok (ngblok is the maximum number of grid points per gridblock, currently set to 640 on Crays & 128 on other machines). |
| **nbcmax** | 1000000 | Maximum memory (in words) used by overlap and kinetic energy integral package, excluding final matrices themselves. |
| **ndisk** | 1500 | Atomic strips of ***J*** and ***K*** are kept in core rather than on disk if (# basis functions) x (# Hamiltonians) < **ndisk**. #Hamiltonians=1 for closed shell and 2 for open-shell. |
| **mxpr** | 100 | Pairs of dealiasing functions are organized so that each group's pairs have the same angular momentum values (e.g. a group with pairs with an s and a p function). The number of pairs in each group evaluated at the same time by subroutine novoro is restricted so that it is ≤ **mxpr**. |

***Table 8.6.33.*** *(Continued) Keywords related to memory and disk usage.*

| keyword | default | description |
|---|---|---|
| **mxrwr** | 100 | Maximum number of dealiasing functions evaluated at a time in subroutine rwrcalc. |
| **zmpmem** | 1.0 | For LMP2 single-point and gradient code, maximum total size allowed for arrays holding partially transformed integrals on grid is 60 MB x **zmpmem**. |

edge of the code should probably *not* change the variables **mxpr** or **mxrwr**.

Finally, the **iq** keywords allow you to choose when to compute the full least-squares fitting matrix $\boldsymbol{Q}$ from the smaller matrix $\boldsymbol{S}[\boldsymbol{R}^{\dagger}\boldsymbol{w}\boldsymbol{R}]^{-1}$ and whether to store it on disk. Names and default values (in bold italics) for these keywords are indicated in Table 8.6.34. If a grid is used only once per calculation, as the fine, ultrafine

***Table 8.6.34.*** *Keywords to determine when to compute the full least-squares fitting matrix* $\boldsymbol{Q}$.

| keyword | value | description |
|---|---|---|
| **iqcoarse** | 0 | For coarse grid, compute $\boldsymbol{Q}$ on the fly in the program scf |
| | ***1*** | For coarse grid, compute $\boldsymbol{Q}$ in the program rwr and store on disk for later use |
| **iqmedium** | 0 | For medium grid, compute $\boldsymbol{Q}$ on the fly in the program scf |
| | ***1*** | For medium grid, compute $\boldsymbol{Q}$ in the program rwr and store on disk for later use |
| **iqfine** | ***0*** | For fine grid, compute $\boldsymbol{Q}$ on the fly in the program scf |
| | 1 | For fine grid, compute $\boldsymbol{Q}$ in the program rwr and store on disk for later use |
| **iqufine** | 0 | For ultrafine grid, compute $\boldsymbol{Q}$ on the fly in the program scf |
| | ***1*** | For ultrafine grid, compute $\boldsymbol{Q}$ in the program rwr and store on disk for later use |

***Table 8.6.34.*** *(Continued) Keywords to determine when to compute the full least-squares fitting matrix* ***Q****.*

| keyword | value | description |
|---|---|---|
| **iqgrad** | *0* | For gradient grid, compute ***Q*** on the fly in the program scf |
| | 1 | For gradient grid, compute ***Q*** in the program rwr and store on disk for later use |

and gradient grids generally are, setting its **iq*name*** value to 0 saves disk space and costs no cpu time. Setting the **iq*name*** values for other grids to 0 will add some cpu cost, but will save some disk space.

# 8.7. THE gvb SECTION

The **gvb** section, whose interface equivalent is described in Section 3.3, is not keyword based. The section should contain the pair settings, in any order, unless you are using the Lewis dot structure keywords described in the *GVB and Lewis Dot Structure Keywords* subsection of Section 8.6. Each line describing a bond pair should contain three integers, which specify the type of bond (1 for sigma, 2 for pi, 3 for a second pi in a triple bond) and the atom number labels of the two atoms in the GVB pair. Each line describing a lone pair should contain a number identifying the lone pair, followed by the number or atom label of the atom associated with the lone pair, and the same atom number or label repeated once more. Either all or none of the lone pairs on an atom should be specified as GVB lone pairs, and these GVB lone pairs should be identified by consecutive numbers starting with 101. Thus, if the molecule had one lone pair on atom 2 and two on atom 5, the lines describing them would contain the numbers '101 2 2,' '101 5 5,' and '102 5 5,' respectively.

Three more entries may be added onto the ends of all of the lines specifying the pairs; these entries will show up in new input files generated during or after calculations. The first value, if it is present, is either 0 or 1, where a 1 entry indicates that a restricted configuration interaction (RCI) calculation including that pair will be performed. (By default, the pair will not be included in an RCI calculation.) The next two values, if they exist, indicate the CI coefficients for the first and second GVB natural orbitals in each pair. The first coefficient should always be positive, and its magnitude should always be greater than that of the second coefficient, which should always be negative. These coefficients are included in new input files

so that if you restart the calculation with the new input file, the contributions of each GVB natural orbital will be known.

The sample **gvb** section which follows sets a sigma bond pair with RCI on between atom 1 and atom 2 and two lone pairs on atom 1.

```
&gvb
1   1 2  1
101 1 1
102 1 1
&
```

# 8.8. THE lmp2 SECTION

The **lmp2** section, whose interface equivalent is described in Section 3.2, is not keyword based. The section should contain a line for each atom pair describing atoms to be treated at the LMP2 level. Each line describing an LMP2 pair should begin with two atom numbers of labels, which specify the two atoms in the pair. Pairs can be listed in any order.

The sample **lmp2** section which follows instructs Jaguar to treat the atoms listed sixth, ninth, and tenth in the **zmat** section at the LMP2 level and all other atoms at the Hartree-Fock level. Atom 9 is bonded to atoms 6 and 10.

```
&lmp2
6   9
9  10
&
```

You can also use the **lmp2** section of the Jaguar input file to list particular LMP2 pairs and request that they be delocalized over listed atoms. With LMP2 delocalization, the space of correlating virtual orbitals for an LMP2 occupied orbital is extended to include orbitals on nearby atoms.

To delocalize a bond pair on two particular atoms over a space including orbitals on a set of other atoms, add a line to the **lmp2** section listing the atom labels or numbers of the two atoms upon which the bond pair is located by default, followed by the atom numbers or labels of any atoms over which the pair is to be delocalized. Next, set the keyword **idelocv** in the **gen** section to 1 (if you want to treat all LMP2 pairs in the system) or 2 (if you want to perform a "local local" MP2 calculation with only the pairs listed in the **lmp2** section treated at the LMP2 level). For example, the following gen and lmp2 sections request a local local MP2 calcula-

tion with the C2–C3 bond pair delocalized over C1 and C4 (as well as over C2 and C3):

```
&gen
mp2=3
idelocv=2
&
&lmp2
C2 C3 C1 C4
&
```

For QST-guided transition state searches with LMP2 wavefunctions, LMP2 delocalization will automatically be performed over neighboring atoms for any bonds present in one structure and not in another, unless the input file contains the **gen** section keyword setting **idelocv**=0.

# 8.9. THE atomic SECTION

The **atomic** section allows you to specify different calculation properties for different atoms in a molecule. These properties can include basis sets for each individual atom, or atomic masses, a feature that allows isotope calculations. You can also use the **atomic** section to define groups of atoms called "fragments," where each fragment can then be converted to dummy atoms or counterpoise atoms or used to define a part of the system for which you want to compute a numerical Hessian. Restart files may include **atomic** sections as well, in order to keep information about charge fitting or other properties calculated previously.

In addition, **atomic** sections can be used to supply information about transition-metal-containing systems that can then be used to generate high-quality initial guesses for these systems. See *Organometallics and Other Difficult-to-Converge Systems* in section 6.3 for more information on using **atomic** sections in this manner.

## *General Format of the atomic Section*

After the '&atomic' (or '$atomic') line, the **atomic** section should list sets of atomic input values. Each of these sets consists of a list of input types (which could include, for example, the **atomic** keywords for atomic masses and for van der Waals radii for solvation calculations) and lists of corresponding values for these input types for each relevant atom. For instance, in the following **atomic** section:

```
&atomic
atom  mass   vdw2
H1    2.00   1.20
H2    2.00   1.20
atom  vdw2
O     1.55
&
```

the input types are **atom** (indicating the atom label or number), **mass** (indicating the nuclear mass in amu), and **vdw2** (indicating the van der Waals radii for a solvation calculation), and the lines for the atoms H1 and H2 indicate that these atoms have a nuclear mass of 2.00 amu (deuterium) and van der Waals radii of 1.2 Å for solvation purposes, while the line for atom O indicates a solvation van der Waals radius of 1.55 Å for this atom. It is not necessary to list information for atoms which are to be treated in the usual, default manner. Input type labels are case insensitive (for example, "MASS" and "mass" are interpreted the same way), and input types can be listed in any order in an input type list. All entries in a row should be separated by one or more spaces or tabs, but columns do not need to be lined up precisely.

The input type **atom** must be included in every set of atomic input values. The corresponding atom identifiers in the lists following each input type list can be either atom labels (such as "H1" or "O" in the example above) or atom numbers (such as "2" for the second atom listed in the **zmat** input). *Atom label input is case sensitive.*

If an input type from the input type list above an atom's description is not one you want to set for that atom, you may use a "?" or "–" to indicate that the default value should be used. Alternatively, if the last input type or types in the input type list are not relevant for that atom, you may leave the end of the line blank. For instance, either the section

```
&atomic
atom  mass   vdw2
H1    2.00   1.20
H2    2.00   1.20
O     ?      1.55
&
```

or the section

```
&atomic
atom  vdw2   mass
H1    1.20   2.00
H2    1.20   2.00
O     1.55
&
```

has the same result as the first **atomic** section example listed above.

Atoms may be described in more than one set of atomic input values, but if data for the same input type is entered twice for the same atom, the first value will be ignored.

To print an **atomic** section in the job's restart file that contains information for all atoms, not just some, set the output keyword **ip29** to 2. If an **atomic** section exists or if **ip29**=2 in a job's input file, the **atomic** section will be echoed in the output from the program pre.

## *Input Types That Specify Physical Properties*

The **atomic** input types that specify physical properties of atoms are listed and defined in Table 8.9.1. Values for these input types may appear in restart files.

***Table 8.9.1.*** *Some input types for the* ***atomic*** *section.*

| input type | description |
|---|---|
| **isotope** | isotopic number (integer, e.g. 2 for deuterium); overridden by atom's **mass** setting if it exists |
| **mass** | nuclear mass in amu |
| **esp** | ESP fitted point charge (or request to fit charge to dummy atom; see text) |
| **formal** | formal charge (integer value) on atom |
| **multip** | spin multiplicity of atom (or fragment containing atom) |
| **mulk** | Mulliken population |
| **vdw** | van der Waals radii (in Å) for charge fitting |
| **vdw2** | van der Waals radii (in Å) for solvation |
| **cov** | covalent radius in Å (used to determine bonding and other properties) |

The **formal** column is useful for solvation jobs (because the van der Waals radii are adjusted according to the chemical structure found by Jaguar) and for generating an improved initial guess for transition-metal-containing systems (along with the **multip** column). Please see the subsection *Organometallics and Other Difficult-to-Converge Systems* in section 6.3 for more information on using this improved initial guess method.

The **esp** column can be used to tell Jaguar to freeze the charge on an atom to a particular value while fitting charges to other atoms, leave an atom out of charge fitting, or fit a charge to a dummy atom. If the

**esp** column entry for an atom is set to a real number, the atomic charge for that atom will be held fixed to that number during charge fitting. If the **esp** column entry for an atom is set to "n" or "no" (or 0, of course), the atom will not be included in charge fitting. If the **esp** column entry for a dummy atom is "y" or "yes", it will be included in the charge fit.

Several warnings apply to the use of the esp column. First, the esp settings must not be inconsistent with the symmetry used for the rest of the job. Second, you should be careful not to overconstrain the charge fitting job. Third, if you are including any dummy atoms in the charge fitting, it may be advisable to perform the charge fitting in a separate job (based on the restart file), for which the charge fitting grid has been altered to include points around the dummy atom(s) by including a **grid** column in the **atomic** section, with "y" or "yes" entries for the dummy atoms, as described below.

## *Basis, Grid, Dealiasing Function, and Charge Usage for Individual Atoms*

The **basis** input type allows you to specify the basis sets used to treat particular atoms. The string provided to describe the basis set should be chosen from the first column of the table in Section 3.8. Lowercase or uppercase letters can be used. Polarization and diffuse functions can be added by appending *, **, +, or ++ immediately after the basis name. The meaning of these symbols is also described in Section 3.8.

Please note that *if you use an atomic section to specify a different basis set (or sets) for one or more atoms than the basis set used for the other atom(s) in the input, you should not change any basis set assignments if you later restart that job.* For instance, if you ran a job whose input file, mixmol.in, contained an atomic section that specified different basis sets for different atoms, you could generate a new input file (restart file) called mixmol.01.in from the job, but if you used this input file for a second job (restarting the old calculation), you would not be able to change the atomic section at all; otherwise, the program would misinterpret the initial guess specified in mixmol.01.in's **guess** section. (Alternatively, you could delete the **guess** section completely and then change the atomic section however you wanted.)

Three other input types shown in Table 8.9.2 allow you to specify whether to include grid points, dealiasing functions, or nuclear charges for listed atoms. The input values "n", "no", "none", and "only" are not case sensitive. Note that you can use the **atomic** section to specify counterpoise atoms, and that settings in the **atomic**

section take precedence over Z-matrix counterpoise input. In the **atomic** section, counterpoise atoms are indicated by using an entry of "n" in the column entitled "charge" (see Table 8.9.2). Also, note that any other word or letter, such as the "Y" entries that may appear in restart files, indicates that the grid, dealiasing function, or charged particles for that atom are included (the usual default for the grid, daf, and charge input types).

***Table 8.9.2.*** *Input types for listing basis, grid, dealiasing function, and charge information for individual atoms in an* ***atomic*** *section.*

| input type | value | description |
|---|---|---|
| **basis** | n, no, or none | use no basis functions on atom |
| | *basis name* | use basis functions from specified basis set on atom |
| **grid** | n, no, or none | do not include any grid points on atom |
| | only | include grid points on atom, but no basis functions, dealiasing functions, or nuclear charge |
| **daf** | n, no, or none | do not include any dealiasing functions on atom |
| | only | include dealiasing functions on atom, but no basis functions, grid points, or nuclear charge |
| **charge** | n, no, or none | treat atom as a counterpoise atom—do not include nucleus or electrons for "atom" |
| | only | include nuclear charge on atom, but no basis functions, grid points, or dealiasing functions |

## *Defining Fragments*

You can use the **frag** input type in the **atomic** section to specify that all atoms with the same **frag** entry be treated in the same fragment. You can then request that all the atoms in one fragment be treated as dummy atoms or counterpoise atoms, or used as the only atoms for which numerical frequencies will be calculated (where Hessian elements for other atoms are zero).

The default **frag** value for each atom is 0, meaning it is not considered part of any fragment. To assign a group of atoms to the same

fragment, in the **frag** column of the atomic section, enter the same value for each atom.

To treat all atoms in a fragment as counterpoise atoms, make the keyword setting **icpfrag**=*fragno* in the **gen** section of the input file, where *fragno* is the integer in the frag column of the **atomic** section for each of the atoms in question. To treat them all as dummy atoms, make the keyword setting **idelfrag**=*fragno* in the **gen** section. To compute partial frequencies for a particular fragment, make the setting **freqfrag**=*fragno* in the **gen** section of a frequency input file.

# 8.10. THE hess SECTION

If an input file has a non-empty **hess** section, the keyword **inhess** in the **gen** section will be set to 2 automatically, and a Hessian will be read in from the **hess** section. Since for a Hessian $\boldsymbol{H}$, $H_{ij} = H_{ji}$, only the elements with $j \leq i$ are read in, and the program symmetrizes the matrix itself later.

Since the Hessian has dimensions of 3Nx3N, where N is the number of atoms (including dummy atoms), it may be large, so files listing all elements in each row by order of rows could be unwieldy and difficult for the user to read. Therefore, the Hessian is assumed to be presented in blocks composed of five columns each (with the last block possibly having fewer than five columns, if 3N is not a multiple of five). The format used for the **hess** section is the same as that used in GAUSSIAN files or BIOGRAF (.hes) files. All Hessian elements for dummy atoms should be set to 0 (as they are in Jaguar output).

Each set of elements from a block of five columns should be preceded by a line containing one or more arbitrary integer labels; for instance, column labels could be convenient for keeping track of the elements when looking at the **hess** section. All of the elements within a five-column block for which *j* (the column indicator) is less than or equal to *i* (the row indicator) are then read in, one row at a time. Each of these rows containing five or fewer matrix elements starts with an integer which is read first; this integer is not used in the program, but can be used to label the matrix elements for convenience in looking over the file. When the relevant matrix elements from that entire five-column block have been read in, the next block is read in in the same way, until all of the matrix elements for the bottom triangular half of the matrix have been entered.

For example, in the unlikely event that you wished to enter this Hessian:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **11** | 21 | 31 | 41 | 51 | 61 | 71 | 81 | 91 |
| **21** | **22** | 32 | 42 | 52 | 62 | 72 | 82 | 92 |
| **31** | **32** | **33** | 43 | 53 | 63 | 73 | 83 | 93 |
| **41** | **42** | **43** | **44** | 54 | 64 | 74 | 84 | 94 |
| **51** | **52** | **53** | **54** | **55** | 65 | 75 | 85 | 95 |
| **61** | **62** | **63** | **64** | **65** | **66** | 76 | 86 | 96 |
| **71** | **72** | **73** | **74** | **75** | **76** | **77** | 87 | 97 |
| **81** | **82** | **83** | **84** | **85** | **86** | **87** | **88** | 98 |
| **91** | **92** | **93** | **94** | **95** | **96** | **97** | **98** | **99** |

you would need to input the elements from the bottom triangle of the Hessian (shown in bold text above) in the following way:

```
&hess
j
i     11
i     21     22
i     31     32     33
i     41     42     43     44
i     51     52     53     54     55
i     61     62     63     64     65
i     71     72     73     74     75
i     81     82     83     84     85
i     91     92     93     94     95
j
i     66
i     76     77
i     86     87     88
i     96     97     98     99
&
```

where the *i*'s and *j*'s indicate integer labels not actually used by the program. In fact, the lines containing *j*'s can contain more than one integer, as described above.

# 8.11. THE guess SECTION

If an input file has a non-empty **guess** section, the keyword **iguess** in the **gen** section is set to 1, and an initial guess for the wavefunction will be read in from the **guess** section. The coefficients contained in the **guess** section are interpreted as coefficients for functions from the basis set indicated with the label **basgss**, if this label exists. For instance, a **guess** section might begin

```
&guess  basgss=6-31g**
```

If no **basgss** setting is listed, the **guess** section wavefunction's basis set is assumed to be the same as the **basis** keyword setting in the **gen** section.

This next line of the section should begin with a set of coefficients describing the contribution of each function in the basis set to the first molecular orbital, and continue on with similar coefficient sets for each molecular orbital. A single line, whose content is unimportant, should precede each molecular orbital's set of coefficients. If you like, you can use this line to label the molecular orbital for your own convenience.

If you choose to output the occupied orbitals, or occupied and virtual orbitals, from one run and use them in the **guess** section for another run, you must make sure to choose a proper format. From the interface's Orbitals window, whose button is found under the Output heading, you could select occupied orbitals or all orbitals from the What option menu and all elements as f19.15, in list or all elements as f8.5, in list from the How option menu for the original run, as described in section 5.7, and the resulting orbital output could be copied from the output file into the **guess** section of the input file for the next run. Similarly, you could set the relevant orbital output keyword to 4, 5, 9, or 10 in the **gen** section of the first run's input file, as described in the subsection *Orbital Output Keywords* in Section 8.6, and use the resulting output file's orbital output in the next run's input file's **guess** section.

If your input file has a non-empty **guess** section and no **basgss** label, or if **basgss** is set to "non_standard", you should make certain that the initial guess included there is for the same basis set specified by the keyword **basis** in the **gen** section, since poor results will obviously be obtained if the initial guess is described with coefficients corresponding to a different basis set than that actually used. Similarly, the ordering of the basis functions within the set being used must be the same as that used for the ordering of coefficients in the **guess** section.

A sample **guess** section for water with an STO-3G basis set follows. The oxygen is atom 1, and for each molecular orbital, coefficients for the oxygen's 1s, 2s, $2p_x$, $2p_y$, and $2p_z$ orbitals are input. The 1s coefficient for the first hydrogen atom follows, followed by the 1s coefficient for the second hydrogen.

```
&guess  basgss=sto-3g
1: orbital energy = -.20251577D+02
   .99421641D+00  .25847131D-01  .31906711D-02   .88241647D-15
   .26760209D-02  -.55838749D-02 -.55838749D-02
2: orbital energy = -.12575489D+01
   -.23376569D+00 .84444935D+00  .94117884D-01   -.39742456D-17
   .78936818D-01  .15559441D+00  .15559441D+00
3: orbital energy = -.59385470D+00
   .30846088D-09  -.13714419D-08 -.39372414D+00 .21348436D-14
   .46944485D+00  .44922200D+00  -.44922200D+00
```

```
4: orbital energy = -.45973017D+00
   .10403593D+00  -.53816730D+00 .57914834D+00  -.40089482D-14
   .48573263D+00  .29510298D+00  .29510298D+00
5: orbital energy = -.39261707D+00
   .26538042D-15  -.27636653D-14 .26424743D-14  .10000000D+01
   .56164871D-15  .78183836D-15  .26536093D-14
&
```

# 8.12. THE pointch SECTION

The **pointch** section describes the locations and magnitudes of a set of point charges. Up to 20,000 point charges may be used.

Each line of the **pointch** section should contain four real numbers, the first specifying the point charge in atomic units, and the next three specifying its (x,y,z) coordinates in the same units used for the geometry (Angstroms by default, but bohr if the **iunit** keyword in the **gen** section is set to 0 or 2; see the *Geometry Input Keywords* subsection of Section 8.6 for more information).

The sample **pointch** section below puts one point charge of charge +1 at location (0, 0, –0.2) and another of charge –1 at location (0, 0, 0.4).

```
&pointch
 1.0  0  0 -0.2
-1.0  0  0  0.4
&
```

Note that point charges should *not* contribute to the value of the net molecular charge, **molchg**, given in the **gen** section.

If you include a non-empty **pointch** section in the input file for a job, the output from the program pre will include a table of fixed charge information describing the point charges. This table will appear in the output file immediately after the molecular geometry output.

# 8.13. THE efields SECTION

If you would like to calculate wavefunctions or molecular properties in the presence of an electric field, you may use the **efields** section to describe this field. The x, y, and z components of the electric field should be specified, in atomic units, on the same line. The requested properties will then be calculated for the molecule in the presence of this field. The scf output will also include nuclear-electric field and electron-electric field terms.

The convention used in Jaguar for electric fields is to add a term of $\vec{E} \cdot \vec{r}$ to the no-field Fock matrix, where $\vec{E}$ is the electric field and $\vec{r}$ is the electron position. The contribution due to the interaction between the field and each nucleus of position $\vec{r}_i$ and charge $q_i$ is $-q_i(\vec{r}_i \cdot \vec{E})$ .

The **efields** section can contain more than one line, describing several different fields. In that case, the calculations for each given field will be performed in turn. Up to 100 electric fields can be specified.

# 8.14. THE ham SECTION

By using the **ham** section and setting the **gen** section calculation keyword **ihamtyp** to 3, you can specify the exact coefficients used to calculate the electronic energy for open shell calculations. The electronic energy is given by the equation

$$E = \sum_i f_i h_{ii} + \sum_{ij} (a_{ij} J_{ij} + b_{ij} K_{ij})$$

where the sums are over orbitals [20]. The number of electron *pairs* per orbital in each orbital $i$ is indicated by $f_i$, which can be listed in the **ham** section, and the one-electron Hamiltonian for that orbital is given by $h_{ii}$. The terms $a_{ij}$ and $b_{ij}$ are coefficients which can also be specified in the **ham** section, and the $J_{ij}$ and $K_{ij}$ terms are Coulomb and exchange terms for pairs of orbitals $i$ and $j$. Orbitals which have the same $a_{ij}$ and $b_{ij}$ coefficients and number of electron pairs $f_i$ are considered to be in the same shell.

The first line in the **ham** section should indicate the number of core orbitals for the molecule. Next, each shell is described in turn. The first line of each shell description should contain two numbers, the first an integer indicating the number of orbitals in that shell, and the second a real number indicating $f_i$, the number of electron pairs in each orbital of that shell. The next line should contain the $a_{ij}$ terms for any orbital in the shell, where $j < i$ and $j$ is not a core orbital. The last line describing the shell lists all $b_{ij}$ terms for any orbital in the shell, where $j < i$ and $j$ is not a core orbital.

# 8.15. THE orbman SECTION

The **orbman** section allows you to reorder orbitals, or to form linear combinations of orbitals. To do so, you can use the **orbman** section

command 'hfiglcmo' (for Hartree-Fock initial guess/linear combination of molecular orbitals), which manipulates the orbitals immediately after the HF initial guess has been generated. The command is used in the following way:

```
&orbman
hfiglcmo    i, j, α    k, l, β   end
&
```

where *i, j, k,* and *l* are integers indicating the *i*th, *j*th, *k*th and *l*th orbitals before mixing (i.e. $\chi_i$, $\chi_j$, $\chi_k$, and $\chi_l$), and $\alpha$ and $\beta$ are angles (in degrees) indicating the degree of mixing. The command hfiglcmo then mixes the orbitals to form orbitals $\chi_i^{new}$, $\chi_j^{new}$, $\chi_k^{new}$, and $\chi_l^{new}$ according to the following equations:

$$\chi_i^{new} = \chi_i \cos\alpha + \chi_j \sin\alpha$$

$$\chi_j^{new} = \chi_j \cos\alpha - \chi_i \sin\alpha$$

$$\chi_k^{new} = \chi_k \cos\beta + \chi_l \sin\beta$$

$$\chi_l^{new} = \chi_l \cos\beta - \chi_k \sin\beta$$

Note that an angle of 90° permutes the two orbitals, reversing the sign of one.

Each combination operation is performed independently, and the operations are performed in the order they are listed in the **orbman** section. Each rotation involving a previously altered orbital uses the new, transformed orbital generated by the earlier operations. After all manipulations have been specified, the word 'end' should be included.

For UHF wave functions, the syntax is modified slightly, and the alpha and beta spin-orbitals are designated by *hfiglcmoa* and *hfiglcmob*, respectively:

```
&orbman
hfiglcmoa    i, j, α    k, l, β   end
hfiglcmob    p, q, γ    r, s, δ   end
&
```

# 8.16. THE echo SECTION

The **echo** section, when it is included in input files, does not contain anything but its own label:

```
&echo &
```

Its purpose is to signal Jaguar to include a copy (echo) of the input file in the output file. If your input file does not contain an **echo** section, the output file will not contain an echo of the input file.

# 8.17. THE path SECTION

The **path** section allows you to specify the execution path, which determines the order of the Jaguar (or other) programs to be run. If no **path** section exists, Jaguar will use the default path resulting from the settings in other sections of the input file.

The items listed in a path can be either Jaguar programs, Unix commands, or other programs accessible from the executable directory. If a program or command is not accessible from the executable directory, you must specify the full path for that program, with a '/' character at the beginning of the path.

Table 8.17.1 gives a brief description of each Jaguar program.

***Table 8.17.1.*** *Individual programs included in Jaguar.*

| **program** | **description** |
|---|---|
| jexec | driver program for all Jaguar executables (note: inclusion of jexec in path will cause recursive Jaguar calculations) |
| pre | reads and checks input (including path, if any), performs symmetry analysis, and calculates terms dependent on geometry (e.g. nuclear repulsion energy) |
| onee | calculates one-electron integrals and effective core potential (ECP) contribution to one-electron Hamiltonian, when relevant |
| hfig | calculates Hartree-Fock initial guess |
| probe | insures orthogonalization |
| grid | generates grids |

***Table 8.17.1.*** *(Continued) Individual programs included in Jaguar.*

| program | description |
|---|---|
| rwr | generates ***Q*** operators |
| gvbig | calculates GVB initial guess |
| scf | performs self-consistent field calculation |
| rci | performs RCI calculation |
| ch | evaluates electrostatic properties (multipole moments, ESP fitting, Mulliken populations) |
| lmp2dip | calculates dipole moments for LMP2 wavefunctions |
| cpolar | finds polarizabilities and hyperpolarizabilities using coupled perturbed HF method |
| polar | finds polarizabilities and hyperpolarizabilities using finite field method |
| elden | calculates electron density on set of grid points |
| loclmp2 | performs localization of orbitals for LMP2 calculations |
| locpost | performs localization of orbitals after final wavefunction is generated |
| lmp2 | performs local second-order Møller-Plesset perturbation theory calculation |
| der1a, der1b | calculate analytic one- and two-electron first derivatives |
| lmp2der, lmp2gda, lmp2gdb | calculate analytic one- and two-electron first derivative terms for LMP2 wavefunctions |
| nude | calculates numerical second derivatives of energy (as numerical derivatives of the analytical gradient) |
| freq | calculates vibrational frequencies and related properties |
| ira, irb | calculate dipole derivative terms needed for calculation of IR intensities |
| geopt | performs geometry optimization |
| pbf | solves Poisson-Boltzmann equations for solvation calculation |
| solv | performs solvation calculation (using results from Jaguar Poisson-Boltzmann solver) |

***Table 8.17.1.*** *(Continued) Individual programs included in Jaguar.*

| program | description |
|---|---|
| sole | checks solvation energy convergence |
| dsolv | computes solvation-related gradient terms for solvated geometry optimizations |
| post | processes files, output, etc. at end of run |
| timex | checks CPU time for entire run |
| machid | utility program; returns machine information (note: not used in Jaguar calculations) |

The simplest form available for the **path** section is a list of the programs to be run, as in the following example:

```
&path pre hfig grid rwr &
```

It is not actually necessary to list pre in paths, since the pre program will always be run.

If you wish to run additional programs after a standard Jaguar calculation, you can use the word 'path' to indicate the default path, as below:

```
&path path name(s) of additional executable(s) &
```

More complicated paths involve looping over programs until the last Jaguar program in the loop indicates that convergence is reached. The first program in the section of the path to be looped over is preceded by a 'loop' label, and the last is followed by a 'goto' label, where each of these labels is followed by the same character string. Nested loops are also allowed. The following path illustrates a loop which will cause the programs pre, onee, grid, and ig to run once, the series of programs rwr, scf, der1a, rwr, der1b, and geopt to run until the convergence criteria for geometry optimization are satisfied, and the program post to run once.

```
&path pre onee grid hfig loopa1 rwr scf der1a rwr der1b geopt gotoa1 post &
```

If you put a 'jump' label between a 'loop' label and a 'goto' label, where 'jump' is followed by the same character string that follows 'loop' and 'goto' ('jumpa1' for the above path, for instance), the path will jump to the end of the loop after the 'goto' label, and will exit the loop, when the 'jump' label indicates that the convergence criterion for that program is reached.

Note that since loops will only exit when convergence is reached, the program before a 'goto' or 'jump' label must have such a criterion. The three programs which can precede a 'goto' or 'jump' label are scf (when it is being used for solvation runs), geopt, and nude.

Sometimes, you may want a path to include a command of more than one word—for instance, you might want to use the Unix command 'mv *old-filename new-filename*' to rename a file. In that case, you can input the **path** section in such a way that each line is assumed to contain a single command. To input the path this way, you must include the word 'line' after the '&path' (or '$path') label at the beginning of the **path** section.

# 8.18. THE plot SECTION

The **plot** section allows you to generate data on one or more orbitals, the potential, or the density in a form that allows you to plot one of these properties on a cubic grid. In order to request this data, you must include a **plot** section with settings to describe what to plot, the dimensions of the box in which it is to be plotted, and the number of points in each direction in the box. Since the **plot** section requests a plot of information from the initial guess unless a **guess** section is included in the input file, we recommend using a restart file resulting from a completed job if you wish to plot final properties from a job. See section 6.4 for information on restart files.

The **plot** section should contain settings for **iorb1a**, **npts**, **origin**, **extentx**, **extenty**, and **extentz**, and may also contain a setting for **iorb2a**. If the setting for **iorb1a** is –2, density plot information will be generated, while if it is –3, potential plot information will be generated. Otherwise, it dictates which orbitals' plot information will be generated. If it is –1, plot information on all orbitals is written out; if it is a positive number, that orbital and all orbitals up to the value of **iorb2a** will be written out. Therefore, to plot only one orbital, you should set **iorb1a** and **iorb2a** to the same number, while otherwise they describe a range (if **iorb1a** is positive).

The other settings describe the grid for which the plot information is generated. The settings for **origin**, **extentx**, **extenty**, and **extentz** describe the box containing these points. The edges of the box start at the origin and form vectors in the three directions determined by the coordinate values of **extentx**, **extenty**, and **extentz**, whose three coordinates should be separated by commas or spaces. The number of points in each direction in the box is then given by **npts**, another

three-dimensional setting. All quantities should be in atomic units (bohr for distances).

Here is a sample **plot** section that causes a job to generate plot information for orbitals 2 through 5:

```
&plot
iorb1a=2
iorb2a=5
npts=22,22,22
origin=-3.150000,-3.150000,-3.150000
extentx=6.300000,0.000000,0.000000
extenty=0.000000,6.300000,0.000000
extentz=0.000000,0.000000,6.300000
&
```

When the job is run, each type of output requested by the **plot** section shows up in a file whose name depends on <jobname>, the name for the job (for example, "h2o" for a job run from the input file h2o.in), and the type of information being plotted. The file name is <jobname>_density.plt for information for a density plot or <jobname>_potential.plt for a potential plot. Orbital plot information shows up in separate files for each orbital, whose names depend on the four-digit orbital number <orbnum>, which would be 0005 for the fifth orbital, for instance. The orbital file names are of the form <jobname>_a<orbnum>MO.plt; for instance, the tenth orbital from the job h2o would appear in a file called h2o_a0010MO.plt.

A .plt file always begins with an echo of the **plot** section used to generate it. The rest of the lines in the .plt file contain values (of the relevant property) to be plotted on the grid described by the **plot** section. The first line gives the value at the origin; next, the values along the vector described by **extentz** are given. The next values correspond to the grid points given by **extentz** but also displaced from the origin along **extenty**. The loop over the points along **extenty** continues, and the outer loop, generating points for displacements along **extentx** for the above square grids, provides the rest of the points in the file.

# 8.19. NBO SECTIONS

To request a Natural Bond Orbital (NBO) analysis at the end of the Jaguar job, you need to include an **nbo** section in your input file. If the section is empty, as it is here:

```
&nbo &
```

a default NBO analysis is performed. Other options for NBO calculations can also be specified in the **nbo** section or in the **core**, **choose**, and **nrtstr** sections of the Jaguar input file. See the NBO documentation for more details on NBO input and output.

# 8.20. RUNNING MULTIPLE JOBS: jaguar batch

You may often need to run series of Jaguar jobs. For instance, you might want to study the dissociation of a bond by evaluating the molecule's energy at various appropriate bond lengths; scan a potential energy surface; perform a Hartree-Fock-level geometry optimization and then evaluate the energy of the new structure using LMP2 or DFT techniques; or simply run several jobs in sequence, one at a time, to avoid overtaxing your machine. If you need to run consecutive Jaguar jobs, you can use the "jaguar batch" command to generate the input files you need and run the necessary jobs.

To use "jaguar batch", you need a batch input file (whose name should end in ".bat") and at least one template Jaguar input file. The batch input file tells "jaguar batch" how to modify the template input file for each Jaguar job. These modifications can include changes to particular bond lengths and angles of the structure, changes in the wavefunction or job type (such as changing an HF geometry optimization input file to a DFT single-point energy calculation input file), changes in the files or directories used for jobs, and virtually all other settings made in input files. One batch input file can be used to request several different input files, either from one template input file or from several different templates. The "jaguar batch" command then generates the input files and runs the corresponding jobs consecutively.

## *Batch Input File Format*

The lines in a batch input file should have the following general format, where this format example is repeated for each job description:

```
[# <Comment>]
[INDIR = <directory containing template input file(s)>]
[OUTDIR = <directory to which output from jobs will be written>]
[TEMP = <root for temporary storage directories for Jaguar jobs>]
[EXEC = <any executable directory listed by "jaguar versions">]
[FLAGS = <-n, -s, or -t "jaguar run" command line options>]
[OPTIONS = <full description of options to apply to jobs below>]
[OPTIONS + <options to add to any previous OPTIONS line settings>]
[% <Unix_command>]
<template_job_name> [<new_job_name> [<options list>]]
```

In the general format description above, the "[ ... ]" symbols denote optional entries, and entries in "<...>" represent character strings whose formats are described below. Note that if you had a set of input files *jobname1*.in, *jobname2*.in, *jobname3*.in, etc., you could simply generate a batch input file whose lines each listed a job name, then run the jobs by using that file as input to "jaguar batch".

For the template job name (or names, if there is more than one such line in the batch input file), you can either use a specific job name for which an input file exists, or you can enter the string "$JOB". For example, a line with the template job name could look like this:

```
h2o    h2o_dft    idft=22111
```

or like this:

```
$JOB    $JOB_dft    idft=22111
```

The string "$JOB" indicates that "jaguar batch" should run the script one or more times, substituting in various job names for $JOB, where these job names listed at the end of the "jaguar batch" command (see below for more details).

Each line of a batch input file that begins with a template job name describes a single new Jaguar job. For each such job, "jaguar batch" first makes a new Jaguar input file (<new_job_name>.in) by changing the template input file (<template_job_name>.in), which is located in the directory listed in the INDIR line (or, if there is no INDIR line, in the current directory from which "jaguar batch" is being run), so that the new input file includes the changes from the options list for that particular job, as well as from any relevant earlier OPTIONS lines. If a <new_job_name> is not included for that job, <new_job_name> is considered to be the same as <template_job_name>. (If the file <new_job_name>.in already exists, "jaguar batch" will overwrite it, unless you are using the "jaguar batch -r" option described later in this section.) Next, "jaguar batch" runs the Jaguar job with any command line options dictated by the FLAGS line, which lets you provide the -n, -s, or -t command line options described at the end of Section 8.1 to "jaguar run". Temporary files generated during the job are written to the subdirectory <new_job_name> in the directory listed in the TEMP line, and output files are written to the directory listed on the OUTDIR line.

Comment lines beginning with "#" can appear anywhere in a batch input file. Blank lines are also allowed.

Any line beginning with "%" should be followed by a shell command. Such commands will be run in the order in which they appear in the batch input.

The INDIR, OUTDIR, TEMP, EXEC, FLAGS, and OPTIONS lines in a batch input file apply to *all* jobs described beneath them, unless a later line of the same type replaces them. Any INDIR, OUTDIR, TEMP, EXEC, or FLAGS line replaces any earlier setting made by the same type of line, and any of these settings can be reset to their default values with the setting NONE (for instance, "FLAGS = NONE"). An OPTIONS line beginning with "OPTIONS =" replaces any earlier OPTIONS line, but any options listed in an options line beginning "OPTIONS +" just get added to the most recent OPTIONS line (if any).

Options for each Jaguar job can be determined by preceding OPTIONS lines or by an options list appearing in the same line as the name of the template input file and new job name. An options list appearing in the same line as a new job name applies only to that job, while an OPTIONS line applies to any jobs described beneath it, unless either a later "OPTIONS =" line *or* the options list for the job supersedes it. Otherwise, the options listed in either of these types of lines are equivalent and use the same formats. These options can specify any of the following items for the relevant jobs:

- keyword settings in the **gen** section of the Jaguar input file
- paths and names of data files (such as the basis set, grid, or other files)
- sections to remove from the template input file (for example, the **guess** section if you are changing basis sets, or the **gvb** section if you are comparing GVB results to HF, LMP2, or DFT results)
- a substitution of a specified number or character string for one already in the template input file

The format for each of these options and an example of each kind are shown in Table 8.20.1, where entries in <> brackets indicate descriptions of the appropriate character string. Host names cannot be included in any of the paths described in the table. Note that you should avoid using any of the characters "$!\<>? in a substitution pattern.

These options and the other line types in a batch input file are illustrated in the sample files in the next subsection. After the examples, directions on how to submit a batch job follow in the final subsection.

***Table 8.20.1.*** *Options in the batch input file that describe one or more Jaguar jobs to be run with "jaguar batch".*

| change | format | examples |
|---|---|---|
| **keyword setting** | keyword = <new value> (or keyword = NONE to remove a setting) | basis = lav3p**<br>idft = 22111<br>igeopt = NONE |
| **data file path and name** | FILETYPE = <file path & name>, or FILETYPE = NONE to return to default choice for that file type | BASISFILE = /usr/es/my.basis<br>ATOMIGFILE = NONE<br>DAFFILE = NONE<br>GRIDFILE = NONE<br>CUTOFFFILE = NONE<br>GPTSFILE = NONE<br>WAVEFNFILE = NONE |
| **section removal** | RMSECTION = <section name> | RMSECTION = guess<br>RMSECTION = gvb |
| **substitution** | <old pattern>==<new pattern> (note: do not use any of the characters "*$!\<>? in either pattern, and do not use spaces around the '==' operator) | bond==1.5<br>torang==170.0 |

## *Batch Input File Example*

As an example, suppose you have ten different molecules and you want to optimize the geometry of each one at the B3LYP/6-31G* level of theory, and then do two single-point energy calculations on the optimized geometry, one using B3LYP/6-311+G* and the other using LMP2/6-311+G. You can create a batch file that will automate this process. The batch file would read in each molecular geometry from an existing input file, make the necessary keyword changes, and perform the calculations. Here is an example of such a batch file (lines beginning with '#' are comments):

```
# B3LYP/6-31G* geometry optimization
$JOB  $JOB_dft_opt  igeopt=1  basis=6-31g* idft=22111

# remove igeopt setting for the following single-
# point calculations and change basis set to 6-311+G*
OPTIONS = igeopt=NONE  basis=6-311+g*

# run B3LYP single-point calculation
$JOB_dft_opt.01  $JOB_dft_sp

# change level of theory to LMP2 and run single-point
# calculation
$JOB_dft_opt.01  $JOB_lmp2_sp  idft=NONE  mp2=3
```

## *Running "jaguar batch"*

To run "jaguar batch", you can enter a command in the form

```
jaguar batch batchname.bat
```

where *batchname*.bat is the name of a batch input file with no occurrences of the string "$JOB". Otherwise, if the batch input file includes "$JOB" entries (described earlier in this section), the "jaguar batch" command should take the form

```
jaguar batch batchname.bat joblist
```

where *joblist* is a list of job names whose corresponding input files already exist and are to be submitted to the batch input file as template jobs. Each Jaguar input file described in the batch input file is then generated (unless doing so would overwrite an existing input file) and used in a Jaguar run.

The "jaguar batch" command also has several command line options, which are summarized in Table 8.20.2. The -r option lets you dictate

***Table 8.20.2.*** *The "jaguar batch" command line options.*

| option | effect | default behavior |
|---|---|---|
| **-r** | forces "jaguar batch" to skip lines of input requiring it to generate <new_job_name>.in files if files of the same name already exist | "jaguar batch" generates Jaguar input files described by batch input file even if they overwrite previously existing files |
| **-l** | "jaguar batch" lists jobs that would be run if it were called without options, but does not generate any files or run any jobs | "jaguar batch" generates new input files and runs corresponding Jaguar jobs |
| **-s** | "jaguar batch" lists jobs that would be run and shows the contents of the input files that would be generated if it were called without options, but does not generate any files or run any jobs | "jaguar batch" generates new input files and runs corresponding Jaguar jobs |

that "jaguar batch" should not overwrite existing Jaguar input files, while the -l and -s flags permit you to see what "jaguar batch" would do, but do not actually allow it to generate any new input files or run any Jaguar jobs.

# 9. Other Jaguar Files

Jaguar needs certain types of files in order to run a job. An input file must be created, of course, but additional files specifying the basis functions, data for the initial guesses, dealiasing functions, grids, and cutoffs used during a run are generally necessary as well. For most calculations, unless other files are specified, Jaguar uses the files default.basis, default.atomig, default.daf, default.grid, and default.cutoff in the data directory. For many solvation calculations, Jaguar also uses the file default.lewis. All of these files are provided by Schrödinger.

If you want to use other data files than those described above, you have several options. First, you can create a new data directory and put files in it whose names and formats match those described above, then specify that data directory from the interface. Note that in order to be able to list that directory as a possible choice, you will have to edit the jaguar.hosts file you are using for the job so that it includes the directory in a 'datapath' line. (See section 6.1 for information on jaguar.hosts files.) A second option is to edit the input file for the job from outside of the interface, adding or editing "BASISFILE," "ATOMIGFILE," "DAFFILE," "GRIDFILE," "CUTOFFFILE," and/or "LEWISFILE" lines with the paths and names of the files you want to use. See section 8.2 for more details. However, if you specify a.cutoff file called "accurate.cutoff," "quick.cutoff," or "solvent.cutoff," the program will assume you are using an outdated file and will reset the name to "default.cutoff," so be careful about using these names for new files.

The rest of this chapter contains descriptions of the basis, atomic initial guess, dealiasing function, grid, cutoff and Lewis files. Even if you do not plan on creating your own versions of these files, you may

want to skim this chapter if you are curious about the methods used in Jaguar.

# 9.1. THE BASIS SET FILE

The basis sets available for use in Jaguar appear in a file called default.basis, in the standard data directories. Portions of this file are shown in this section; you may wish to refer to them as you read the description of the file.

The basis sets are described in turn. Basis sets at the top of the file do not contain effective core potentials, and will be described first here. The basis sets with effective core potentials, whose names begin with "LA," will be described later.

Each basis set description begins with a blank line. The next line (or lines) must begin with the word 'BASIS,' followed by one space. That label is followed by one or more names of the basis set to be described: the name of the basis set as given in Table 3.8.1 or Table 3.8.2 in Section 3.8, and any other names which describe the same basis set (e.g. STO-3G and STO3G). The basis set names are separated by commas, and must include '*' and/or '+' characters, if those are allowed for that basis set. ('**' or '++' character strings are sufficient to describe the '*' and '+' cases also, and the '*' characters can be listed either before or after the '+' characters.) The next notation in the line, '5D' or '6D,' sets the default number of functions for d shells when using that basis set, as described in Section 3.8.

"Backup" basis set name or names, which are each preceded by the word 'BACKUP,' may follow on the same line. If any sets are listed after the word 'BACKUP,' it indicates that if an atom is not found in the current basis set (whose name is listed before 'BACKUP'), its basis function will be obtained from the list of backup basis sets. (If there is more than one backup name listed, the basis function for the atom will come from the first backup set listed that contains that atom.) Note that the numbers of d shells specified in the backup basis sets are ignored. Also, polarization or diffuse functions are chosen according to the basis set specified by the calculation; that is, *, **, +, or ++ options on backup basis sets are ignored if they do not agree with the options on the basis set chosen for the calculation.

The basis set description continues with a set of lines describing the basis functions on each atom. The information for each atom begins with a line containing the element's symbol (e.g. He). The atomic symbol must not be preceded by any spaces or characters. The next

line begins with the type of function being described (S, P, or D, for instance). If this label is 'SP,' the corresponding set of data describes an s *and* a p function whose Gaussians have the same exponents. The next number in that line is the polarization/diffuse function parameter. If it is a 1, it indicates a polarization function which is included in the basis set if the basis set name ends in a *, as described in Section 3.8. If the number is a 2, it indicates a '**' basis set function; if –1, a '+' basis set function; if –2, a '++' basis set function. Otherwise, the number should be 0.

The rest of the numbers on that line determine the way that Jaguar will contract some of the functions, and the "range" of each function. The numbers before the dash (–) describe how many of the functions are included in that contraction. For example, if there were two such numbers, 2 and 1, the line would indicate that Jaguar would contract the first two Gaussians provided immediately below into one contracted function, and would treat the third Gaussian as an uncontracted function.

If you want to add or change a basis set to a .basis file, you should probably contract together all Gaussians whose exponents are greater than 0.3. The default.basis information generally follows this rule, although there are some exceptions (see the Li s and p function information in the sample file below for an example).

The numbers after the dash describe the range of each such function. There should be one such number for each contraction number before the dash. A zero indicates that the contracted function will be treated as a long-range function, while a 1, 2, 3, or 4 indicate various types of short-range functions. These assignments help determine the symmetrization of the Fock matrix components by the "side choosing" method described in reference [13]. These range values are only used in pseudospectral calculations, so if your basis set will be used for non-pseudospectral calculations, just use a '0' as a place holder for each range value. (Pseudospectral calculations require that grids and dealiasing functions exist for the basis set. These will be defined in the default.grid and default.daf files, respectively; see below.)

The Gaussians in the contraction are listed next, with the first number in each of these lines describing the exponent for the Gaussian, and the second its coefficient in the contraction. The Gaussians should be listed in decreasing size of exponent. If both s and p functions are being described, the second number on the line corresponds to the coefficient for that Gaussian in the s function's contraction, and the third number corresponds to the p function's contraction coefficient.

The data for that atom ends with a line containing 4 * characters, with no spaces or other characters preceding them.

When all of the atoms for that basis set have been listed, ending with the obligatory '****' line, the next basis set is listed, in the same manner described above.

The beginning of the default.basis file is shown below to illustrate most of these points.

```
BASIS STO-3G*,STO3G*,STO-3*,STO3* 5D
H
S    0       2  1   -   1  0
     3.42525091400000         0.154328967294599
    0.623913729800000         0.535328142281266
    0.168855404000000         0.444634542184440
****
He
S    0       3   -   2
     6.36242139400000         0.154328967291452
     1.15892299900000         0.535328142270350
    0.313649791500000         0.444634542175373
****
Li
S    0       3   -   4
     16.1195747500000         0.154328967293323
     2.93620066300000         0.535328142276839
    0.794650487000000         0.444634542180763
SP   0       1  2   -   1  0
    0.636289746900000        -9.99672291865986E-02    0.155916274998087
    0.147860053300000         0.399512826086407        0.607683718592546
    4.808867839999999E-02     0.700115468876179        0.391957393095192
****
```

Basis sets containing effective core potentials (ECPs) are described in a slightly more complicated fashion. First, the string 'ECP' must appear between the '5D' or '6D' label and the 'BACKUP' label. This string indicates that the basis set description contains information about the effective core potential associated with the basis set.

As for the basis sets without effective core potentials, each atom in the set is described in turn. The description begins with the basis function, which is in the same format as those described above. After a line with two asterisks ('**'), the effective core potential is described.

The first line in the effective core potential description contains the element symbol (e.g. Na) and two numbers. The first of these indicates the maximum angular component in the core, and the second indicates the number of electrons replaced by the effective core potential. Next, the information for various angular components is

listed. The first set of such information contains the local components of the ECP and should begin with a line saying "D_AND_UP," "F_AND_UP," or "G_AND_UP," which indicates that the maximum angular components to be described are 2, 3, or 4, respectively.

Following that line, the different terms for this angular component are given. Each line describes a term of the form $Ce^{-\alpha r^2}r^{n-2}$, listing the parameters $n$, $\alpha$, and $C$ (from left to right) in a free format. Next, the lower angular components are listed in increasing order (S, P, D, etc.) in the same fashion.

A line with four '*' characters appears the end of the description of each atom's ECP. When all atoms have been described in turn, the next basis set is described.

The example below shows the beginning of the description in default.basis of the LAV2D and LAV2P basis sets. Note that these basis sets only differ in their choice of what basis set to use for atoms which are *not* described by the effective core potential.

```
BASIS LAV2D**,LANL1DZ** 5D ECP BACKUP D95V**
BASIS LAV2P** 5D ECP BACKUP 6-31G**
Na
S    0       1  1   -   1  0
    0.497200000000000       -0.275357400000000
    5.600000000000000E-02     1.09899690000000
S    0       1   -   0
    2.210000000000000E-02     1.00000000000000
P    0       1  1   -   1  0
    0.669700000000000       -6.838450000000000E-02
    6.360000000000000E-02     1.01405500000000
P    0       1   -   0
    2.040000000000000E-02     1.00000000000000
**
Na   2  10
D_AND_UP
1   175.55025900   -10.00000000
2    35.05167910   -47.49020240
2     7.90602700   -17.22830070
2     2.33657190    -6.06377820
2     0.77998670    -0.72993930
S-D
0   243.36058460     3.00000000
1    41.57647590    36.28476260
2    13.26491670    72.93048800
2     3.67971650    23.84011510
2     0.97642090     6.01238610
P-D
0  1257.26506820     5.00000000
1   189.62488100   117.44956830
2    54.52477590   423.39867040
2    13.74499550   109.32472970
```

```
2     3.68135790    31.37016560
2     0.94611060     7.12418130
****
```

If you want to set up your own .basis file, you can do so, if you use the format described above. Generally, *you must also create an altered version of the .atomig file*, which is described in Section 9.2, although if you are just adding polarization functions to the basis set, and these functions are identified by the polarization/diffuse function parameter described earlier in this section, you can continue to use the usual .atomig file. Make sure your new .basis file contains the 6-31G basis set, because the initial guess program needs this basis set. If you alter the basis functions in the default.basis file only slightly, you can use the same names for the basis sets. If you change them a great deal, you should use a new name, so that Jaguar will not attempt to use grids or dealiasing functions that do not match the new basis set. If you change a basis set name to something Jaguar does not recognize, runs using that basis set will use all-analytic methods (see the subsection *Analytic Corrections* in Section 3.9, or the information on the input file **gen** section keyword **nops** in the subsection *Keywords for SCF Methods* in section 8.6).

To use the file in a Jaguar calculation, you must add a line in the form

```
BASISFILE: <basis file path and name>
```

to the input file for the job. You can specify a file on another host, or under another account name on that host, by listing the file name in the format <host>:<file path and name on host>, or <user>@<host>:<file path and name on host>.

# 9.2. THE INITIAL GUESS DATA FILE

The file default.atomig contains the results of Hartree-Fock calculations on atoms for various basis sets. By default, the initial guess is constructed from wavefunctions in this file. When the basis set to be used for the calculation is 6-31G, MSV, LAV2P, LAV2D, LAV3P, LAV3D, LACVP, or LACVD (or any variant of these sets involving polarization or diffusion functions (e.g. 6-31G*)), the initial guess is formed from the wavefunctions obtained from individual calculations on the atoms in the molecule which were calculated using that same basis set (ignoring polarization and diffusion functions). *Therefore, if you change the .basis file, you need to change the .atomig file correspondingly, and vice versa.*

For other basis sets, the wavefunctions used to construct the initial guess are obtained by projecting either the appropriate atomic wavefunction in default.atomig onto the basis set actually being used for the molecular calculation. The 6-31G wavefunction is used whenever possible; when a 6-31G atomic wavefunction is not listed for a particular atom, the MSV one is used for that atom. For atoms beyond Xe in calculations using the LAV1S basis set, the LAV2P atomic results are used.

As in the default.basis file, the basis sets listed in the default.atomig are listed in turn, and for each basis, the information for each atom is listed. Each basis set section begins with a blank line, which is followed by one or more lines reading 'BASIS,' followed by one space, and ending with the name or names (separated by a space and/or comma) of all basis sets for which the atomic calculations listed immediately after that line apply. The basis set names are listed in Table 3.8.1 and Table 3.8.2 in Section 3.8.

Next, the information for each atom follows. The first line lists the atomic symbol for the atom, followed by information which is simply a comment and is not read in. The next line lists two numbers. The first of these numbers gives the number of basis functions for that atom and basis set, as listed in the default.basis file, and the second gives the number of electrons for that atom included in an effective core (0 for the basis sets whose names do not start with "LA"). The line after that lists the orbital number (1 if it is the first orbital listed for that atom, 2 if it is the second, and so on), the orbital occupation (i.e, the number of electron pairs in that orbital), and the orbital energy in Hartrees. That orbital's coefficients for each basis function for the given atom and basis set(s) follow on the next line(s).

When all of the orbitals for that atom have been specified, a line with 4 '*' characters indicates the end of the information for that atom, and the data for the other atoms is listed. Similar information for each other basis set follows.

If you want to set up your own .atomig file, you can do so, if you use the format described above. To use the file in a Jaguar calculation, you must add a line saying

```
ATOMIGFILE: <atomic initial guess data file path and name>
```

to the input file for the job. You can specify a file on another host, or under another account name on that host, by listing the file name in the format <host>:<file path and name on host>, or <user>@<host>:<file path and name on host>.

# 9.3. THE DEALIASING FUNCTION FILE

When Jaguar fits a function's grid point values to a basis set to find the applicable basis set coefficients for the function, it uses dealiasing functions to reduce errors. The dealiasing functions span the function space determined by the grid more completely than the basis functions, so a function on the grid can be better described using the dealiasing functions than by the basis functions alone. The basis set coefficients for the function can then be determined by using the overlap between the dealiasing functions and the basis set functions, which is determined analytically.

Some basis functions die off slowly and require long-range functions centered on each atom in the molecule, while others die off quickly over distance and can be described with short-range dealiasing functions centered on the nearby atoms. The latter type can employ different dealiasing functions, depending on the distance between the atom upon which the relevant basis function is centered and the atom upon which the short-range dealiasing functions are to be centered. If the atoms are the same, "home atom" dealiasing functions are used; otherwise, the distance between the two atoms determines whether the dealiasing functions used should be those for first-order or one of the other higher-order neighbors. (To see this connectivity information for a system, use the **gen** section keyword setting **ip12** = 2.) If the two atoms are further away than the farthest neighbor range specified, no dealiasing functions on one atom are used in calculating the contribution of a short-range basis function centered on the other atom.

The dealiasing functions themselves are simple polynomials multiplied by Gaussian functions, and are s-type, p-type, and so on, depending on the polynomial. Uncontracted dealiasing functions are simply formed by specifying the exponent of the Gaussian function. Contracted dealiasing functions are defined as linear combinations of the appropriate type of functions; the coefficients and exponents for the linear combination are the same as those used in the basis set for the contracted basis functions for the relevant function types (1s, 2s, $2p_x$, etc., depending on the molecule and the basis set). Thus, a dealiasing uncontracted function can be specified by dictating the type (s, p, d, etc.) and the exponent desired for the Gaussian, while a contracted Gaussian function can be specified by dictating the type and referencing which set of contraction coefficients and exponents are desired.

The *File Format and Description* subsection below describes the file that determines the dealiasing functions for a calculation. Sets of

dealiasing functions must be provided for each grid used in the calculation. Comments about a sample file refer to the sample .daf file in the subsection *Sample File*.

## *File Format and Description*

The first line of a dealiasing function file contains a character string which includes the version number of Jaguar. This string should be "dafv" followed immediately by four digits giving the version number times 100. Lead zeros are added if necessary.

The next line is made up of two integers. The first integer dictates the number of dealiasing function sets provided for each atom type; each set is used for a particular grid during the calculation. The ordering of the sets used for each grid type is determined by the parameters named **dcoarse**, **dmedium**, and so on, which are specified in the **gen** section of the input file. (By default, the coarse grid is listed first, then the medium, fine, ultrafine, and gradient grids, in that order.)

The second number in the second line gives the number of ranges described in each of these dealiasing function sets. The ranges correspond to particular RwR blocks for the calculation. One of these ranges is the long range, basically covering the whole molecule; another is the home atom range, which actually only includes the relevant atom itself; and the rest are increasingly large neighbor ranges. The number of ranges should currently not exceed 10. The sample file's second line indicates that for each basis set, five dealiasing function sets are specified for each atom, and that each of these sets contains dealiasing functions for a total of six ranges: the long-range functions, the functions for the home atom, and the functions for four other neighbor ranges.

The distances defining the neighbor ranges are set in the next line of real values, in units of bohr. (Please note, however, that generally only the third neighbor range is actually used.) The first distance specifies that if the basis function whose coefficient is being evaluated is to be approximated by short-range dealiasing functions, then the dealiasing functions for first-order neighbors will be used for each atom within this distance of the atom upon which the basis function is centered (except for the basis function atom itself, for which the home atom dealiasing functions will be used). The second distance defines which atoms are considered second-order neighbors to each other, and so on. Since the number of neighbor ranges includes not only these ranges but also the long range over the entire molecule and the home atom range consisting of the relevant atom itself, the number of neighbor ranges actually specified in this line of the .daf file should be two less

than the number of ranges listed in the previous line. Thus, in the sample file, the distances listed specify the neighbor ranges for first- through fourth-order neighbors.

The rest of the .daf file contains the dealiasing function sets for each atom type within each basis set. The data for each basis set should begin with a line listing the basis set name (as listed in Table 3.8.1 and Table 3.8.2 in Section 3.8), including the '*' characters indicating the polarization functions (e.g. 6-31G**). The first line for each atom type for that basis set should list three integers: the atomic number for that atom type, the number of uncontracted dealiasing functions about to be listed for each neighbor range in each set, and the corresponding number of contracted dealiasing functions. In the sample file, the first atom whose dealiasing functions are listed is hydrogen, since the atomic number listed is 1. The same line says that ten uncontracted functions and two contracted functions will be specified for each range in the five sets of dealiasing functions for hydrogen.

The second line for the same atom type should list real dealiasing exponents for each uncontracted function. The exponents specify what functions can be used. For instance, in the sample file, hydrogen's s-type uncontracted basis function from the first exponent would be $N_1 e^{-.040634 r^2}$, while the p-type uncontracted basis function for the same exponent would be $N_2 r e^{-.040634 r^2}$. (The $N$'s are normalization constants.)

Beneath those two lines, the dealiasing function sets for that atom type should be listed set by set. By default, the first set will be used for the coarse grid, the second for the medium grid, and so on, with the last set corresponding to the gradient. This ordering can be changed in the **gen** section of the input file. Each set should contain a line for each neighbor range; the long-range functions should be specified first, then the home atom functions, then the functions for each neighbor range, in increasing order. Within each line, there should be several integers, one for each uncontracted function, then one for each contracted function. These integers dictate how to construct the actual functions from the exponents (just given in the .daf file for uncontracted functions, and already established in the .basis file for contracted functions) and contraction coefficients for contracted functions (also established in the .basis file). If the value is 1, an s-type function will be constructed using the relevant exponent or exponents; if 2, a p-type function; if 4, a d-type function; if 8, an f-type function; and if 16, a g-type function. To construct more than one of these types of functions with the same exponent or exponents, the relevant numbers should be added together (for instance, 1 + 2 + 4 = 7 for s, p, and d).

The exponent or exponents for each of these functions will be determined by the position of the entry in the row. The uncontracted functions are described first, in the same order as their exponents were listed earlier, and the contracted functions corresponding to the *contracted* functions found in the .basis file are described next, in the same order as in the basis set file. (Uncontracted functions in the basis set file should be ignored.) Finally, the first derivatives of the basis set file contracted functions will be calculated, and the values listed for these "extra" functions will correspond to the functions generated this way, in order of the function they were generated from and, within that order, of increasing complexity (s before p, etc.). For instance, if the basis set contained contracted functions for 1s, 2s, and 2p orbitals, the derivatives would be listed in the following order: a p-type function resulting from the derivative of the 1s function, a p-type function resulting from the derivative of the 2s function, an s-type function resulting from the first term of the derivative of the 2p function, and a d-type function resulting from the second term of the derivative of the 2p function.

The last six lines of the sample .daf file correspond to the gradient dealiasing function set for He (note that the atomic number specified for those five dealiasing function sets was 2). The first line of this set describes this set's long-range dealiasing functions centered on the He atom, which will be used when coefficients for long-range basis functions are to be calculated, as explained above. The second value on this line, 3, dictates that uncontracted s-type and p-type (1 + 2 = 3) basis functions are to be constructed using the second exponent provided for this atom (0.145957). The second line of the set, which describes this set's He-centered dealiasing functions to be used when calculating the coefficients for He-centered short-range basis functions (the home atom line of the set), has a value of 1 entered in the eleventh column, meaning that an s-type contracted function will be calculated using the exponents provided for the first contracted function for He in the basis set. Since this basis set only provides one contracted function for He, the 1s function, whose derivative is a p-type function, the last number entered on that line (2) dictates that a p-type function be constructed, using the contraction coefficients and exponents that correspond to that derivative function, as explained in the previous paragraph.

## *Sample File*

The following sample .daf file lists the dealiasing sets for H and He for a 6-31G** basis set. Note that since a FORTRAN free-format read statement reads lines until all variables are assigned values,

blank lines may be added for readability, and data may be spread over multiple lines.

```
dafv0300
5  6              <-- number of sets/atom, number of rows/set
3.0 5.0 7.0 9.0  <-- neighbors cutoffs distances (neighbors = no_of_row - 2)

BASIS 6-31G**

 1 10 2 (H)
0.040634 0.080953 0.161278 0.321306 0.640122 1.275283 2.540684 5.061679
10.084136  1.100000

 0  3  3  3  0  0  0  0  0   0   0  0
 0  0  7  0  0  5  0  0  0   2   1  0
 0  0  7  3  2  5  0  0  0   2   1  0
 0  0  3  0  2  1  0  0  0   2   1  0
 0  0  0  0  0  0  0  0  0   0   0  0
 0  0  0  0  0  0  0  0  0   0   0  0

 0  3  3  3  0  0  0  0  0   0   0  0
 0  0  7  0  0  5  0  0  0   2   1  0
 0  0  7  3  2  5  0  0  0   2   1  0
 0  0  3  0  2  1  0  0  0   2   1  0
 0  0  0  0  0  0  0  0  0   0   0  0
 0  0  0  0  0  0  0  0  0   0   0  0

 0  3  3  3  0  0  0  0  0   0   0  0
 0  3  7  0  0  5  0  0  0   2   1  0
 0  3  7  3  2  5  0  0  0   2   1  0
 0  0  3  0  2  1  0  0  0   2   1  0
 0  0  0  0  0  0  0  0  0   0   0  0
 0  0  0  0  0  0  0  0  0   0   0  0

 0  3  3  3  0  0  0  0  0   0   0  0
 0  3  7  0  0  5  0  0  0   2   1  0
 0  3  7  3  2  5  0  0  0   2   1  0
 0  0  3  0  2  1  0  0  0   2   1  0
 0  0  0  0  0  0  0  0  0   0   0  0
 0  0  0  0  0  0  0  0  0   0   0  0

 0  3  3  3  0  0  0  0  0   0   0  0
 0  3  7  0  0  5  0  0  0   2   1  2
 0  3  7  3  2  5  0  0  0   2   1  2
 0  0  3  0  2  1  0  0  0   2   1  2
 0  0  0  0  0  0  0  0  0   0   0  0
 0  0  0  0  0  0  0  0  0   0   0  0

 2 10 2 (He)
0.071497 0.145957 0.297964 0.608279 1.241774 2.535023 5.175131 10.564786
21.567514  1.100000

 0  3  3  3  0  0  0  0  0   0   0  0
 0  0  7  0  0  5  0  0  0   2   1  0
 0  0  7  3  2  5  0  0  0   2   1  0
 0  0  3  0  2  1  0  0  0   2   1  0
 0  0  0  0  0  0  0  0  0   0   0  0
 0  0  0  0  0  0  0  0  0   0   0  0
```

```
0   3   3   3   0   0   0   0   0   0   0   0
0   0   7   0   0   5   0   0   0   2   1   0
0   0   7   3   2   5   0   0   0   2   1   0
0   0   3   0   2   1   0   0   0   2   1   0
0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0

0   3   3   3   0   0   0   0   0   0   0   0
0   3   7   0   0   5   0   0   0   2   1   0
0   3   7   3   2   5   0   0   0   2   1   0
0   0   3   0   2   1   0   0   0   2   1   0
0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0

0   3   3   3   0   0   0   0   0   0   0   0
0   3   7   0   0   5   0   0   0   2   1   0
0   3   7   3   2   5   0   0   0   2   1   0
0   0   3   0   2   1   0   0   0   2   1   0
0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0

0   3   3   3   0   0   0   0   0   0   0   0
0   3   7   0   0   5   0   0   0   2   1   2
0   3   7   3   2   5   0   0   0   2   1   2
0   0   3   0   2   1   0   0   0   2   1   2
0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0
```

# 9.4. THE GRID FILE

The grid input file (.grid file) determines the grids used during the calculation. Each grid type, for example, "coarse" or "ultrafine," is constructed from grids assigned to each atom in the molecule. Therefore, for any basis set for which the pseudospectral method is used, the grid file must contain grids for each grid type used, where each of these grid types in turn requires atomic grids for each element in the molecule. The assignment of grids to grid types is performed within the input file, using the **gen** parameters **gcoarse**, **gmedium**, and so on.

## *File Format and Description*

The first line of a .grid file contains a character string which includes the version number of Jaguar. This string should be "gridv" followed immediately by four digits giving the version number times 100. Lead zeros are added if necessary.

The next line should consist of an integer which gives the number of grid types described in the file. For instance, this number would be six

if the grids specified were of the types coarse, medium, fine, ultrafine, eldens (for electron density calculations), and gradient. By default, Jaguar uses the coarse grid for electron density calculations and the ultrafine grid for gradient calculations, and the "extreme" grid is included for testing purposes, so the number of grid types in the file default.grid is actually five. Jaguar uses the grids upon each atom in the molecule provided by the .grid file to generate molecular grids.

All grids for each basis set are then listed in turn. The basis set is identified with a BASIS line and containing its name, and is followed by a blank line.

Each molecular grid description starts with two comment lines, usually a blank line followed by a descriptive line. The next line contains an integer flag which determines which points from the atomic grids for the atoms in a molecule are included in the molecular grid. Jaguar generates a boundary plane between the two atoms and perpendicular to the vector between them, disposing of any points from one atom that are on the other atom's side of the boundary plane. The integer flag determines the location of this plane: if the flag is 0, the plane is located so that the ratio of the distances of the atoms to the plane is the same as the ratio of their covalent radii, while if it is –1, the boundary plane is set where the grid point density from each atom, on the vector between the atoms, is equal. The grid point density is determined as a spline fit of the density for each shell, where each shell's density is determined as the number of points for that shell divided by the shell volume, which is the volume between the spheres whose radii are the average of the current and previous shell radii, and the current and following shell radii.

After the flag for the grid, information for each atomic grid is provided. The first line of each atomic grid section contains two integers, one providing the atomic number for that atom and the other giving the number of shells to be described. Currently, this second number should be 30 or less. The next line contains that number of entries defining the radial shell spacing, listing the radius of each shell in bohr. Grid points for that shell will be placed at that radius, in a pattern determined by the integers given in the third line. This last line of integers represents the density of the angular grid for each shell. The values are explained below.

The default.grid file for Jaguar version 4.2 begins as follows:

```
gridv0220
5 24

BASIS 6-31G
```

```
coarse grid
-1

1 6
0.23021 0.71955 1.74518 2.82595 3.94135 6.40743
1 3 7 7 3 1

2 7
0.20699 0.45860 0.97184 1.61794 2.40119 3.26487 5.20964
1 3 7 9 7 3 1

3 7
0.59584 1.69094 3.39571 5.30494 7.49262 11.30338 16.61803
1 3 7 9 7 3 1
```

Since a FORTRAN free-format read statement reads lines until all variables are assigned values, blank lines have been added between atomic grids for readability. Data may be spread over multiple lines.

As explained above, the beginning of the default.grid file indicates that five grid types are listed for each atom (corresponding to the coarse, medium, fine, ultrafine, and gradient grids. All coarse grids for 6-31G (with or without the polarization functions indicated by the '**') will set the boundary plane between atoms (described earlier) at the point where the grid point densities are the same for the two atoms, because of the '–1' flag. Next, seven shells apiece are specified for H (atomic number 1), He (atomic number 2), and Li (atomic number 3). The actual default.grid file continues with a list of coarse atomic grids for the other atoms in the basis set, followed by the medium, fine, and ultrafine atomic grids in the same format, before proceeding to define the grids for another basis set in the same manner.

The possible values of the numbers on the angular grid line are listed in Table 9.4.1, along with the corresponding number of points per

***Table 9.4.1.*** *Number of points per angular shell and degree of the highest spherical harmonic exactly integrated by grids specified by various entries on the angular grid line.*

| entry | points | degree | reference for grid |
|---|---|---|---|
| 1 | 6 | 3 | Un 3-1 (Stroud), p.294 [110] |
| 2 | 8 | 3 | Un 3-2 (Stroud), p.294 [110] |
| 3 | 12 | 3 | U3 3-1 (McLaren), p.296 [110] |
| 4 | 14 | 5 | Un 5-2 (Albrecht & Collatz), p.294 [110] |
| 5 | 18 | 5 | Un 5-1 (Albrecht & Collatz), p.294 [110] |

***Table 9.4.1.*** *(Continued) Number of points per angular shell and degree of the highest spherical harmonic exactly integrated by grids specified by various entries on the angular grid line.*

| entry | points | degree | reference for grid |
|---|---|---|---|
| 6 | 18 | 5 | Un 5-1 (Albrecht & Collatz), p.294 [110] |
| 7 | 24 | 5 | Un 5-4 (Stroud), p.295 [110] |
| 8 | 26 | 7 | Un 7-1 (Albrecht & Collatz), p.295 [110] |
| 9 | 38 | 9 | 9.1 (Lebedev) [111] |
| 10 | 38 | 9 | 9.1 (Lebedev) [111] |
| 11 | 42 | 9 | 9.2 (Lebedev) [111] |
| 12 | 44 | 9 | 9.3 (Lebedev) [111] |
| 13 | 44 | 9 | 9.4 (Lebedev) [111] |
| 14 | 50 | 11 | U3 11-1 (McLaren), p.301 [110]; 11.1 (Lebedev) [111] |
| 15 | 54 | 11 | 11.2 (Lebedev) [111] |
| 16 | 56 | 11 | U3 11-2 (Stroud), p.301 [110] |
| 17 | 60 | 11 | 11.3 (Lebedev) [111] |
| 18 | 60 | 11 | 11.3 (Lebedev) [111] |
| 19 | 78 | 13 | 13.2 (Lebedev) [111] |
| 20 | 78 | 13 | 13.3 (Lebedev) [111] |
| 21 | 86 | 15 | 15.1 (Lebedev) [111] |
| 22 | 90 | 15 | 15.2 (Lebedev) [111] |
| 23 | 90 | 15 | 15.2 (Lebedev) [111] |
| 24 | 110 | 17 | 17.1 (Lebedev) [111] |
| 25 | 116 | 17 | 17.2 (Lebedev) [111] |
| 26 | 146 | 19 | 19 (Lebedev) [112] |
| 27 | 146 | 19 | 19 (Lebedev) [112] |
| 28 | 194 | 23 | 23 (Lebedev) [112] |
| 29 | 302 | 29 | 29 (Lebedev) [113] |
| 30 | 434 | 35 | Lebedev [114] |

angular shell and the degree of the highest spherical harmonic which the grid integrates exactly, when relevant. (The full references are provided near the end of the User's Guide.)

# 9.5. THE CUTOFF FILE

The cutoff file specifies parameters to be used for the various iterations of an SCF calculation. The file to be used is determined by the 'CUTOFFFILE' entry in the input file, as described in section 8.2; if the input file has no such line, Jaguar uses the file default.cutoff from the data directory. Note that if the 'CUTOFFFILE' entry is "accurate.cutoff," "solvent.cutoff," or "quick.cutoff," the program will interpret the setting as "default.cutoff."

The first line of a cutoff file contains a character string which includes the version number of Jaguar. This should be "cutv" followed by four digits giving the version number times 100. Lead zeros are added if necessary. A comment on the same line can follow the version string.

The next five lines each have five numbers. Each line describes a particular level of accuracy to be used for the calculation. The first line provides the information necessary to run a calculation with all-ultrafine pseudospectral grids and with "tight" cutoffs, and corresponds to an accuracy level setting of ultrafine from the interface, as described in Section 3.9 under *Accuracy Level*, or to the keyword setting **iacc** = 1 in the **gen** section of the input file, as described in the *Keywords for SCF Methods* subsection in section 8.6. The second line gives the parameters for the accurate level (**iacc** = 2), while the third line provides information for the quick level (**iacc** = 3). The last two lines are filled with zeroes, since they need to be present, but are not yet used.

In each of these rows, the columns describe which cutoff sets are used for various SCF iterations. The cutoff sets themselves are provided later in the file, and dictate the level of analytic "corrections," the grid, and the non-default values of the **gen** section cutoff keywords (**cut1**, for example). The cutoff sets are described in more detail below. The columns reflect a scheme in which calculations are broken down into preliminary and final sets of iterations. The iterations from the beginning of the first SCF calculation in a run are considered to be part of the preliminary set, while the iterations from the end of the first SCF calculation, or from any subsequent set of SCF iterations, are considered to be part of the final set. For instance, for a solvation calculation, the SCF iterations for the analysis of the converged gas phase wavefunction are preliminary iterations followed by final iterations, while the SCF iterations for all subsequent SCF calculations, those including the solvent effects, are final iterations. Jaguar determines how many iterations are preliminary and how many are final for the initial SCF calculation.

The number in the first column in each of the five accuracy level lines dictates the cutoff set used for the first iteration in the preliminary sequence: if the number is a 1, the first cutoff set listed in the file is used; if it is a 5, the fifth is used, and so on. The number in the second column provides the cutoff set used for updates during the preliminary sequence of iterations. The third and fourth columns describe the cutoff sets used for the first and updating iterations in the final sequence, respectively. Finally, the last column dictates the cutoff sets used for non-SCF calculations, as for gradient calculations.

The first six lines of the default.cutoff file, which illustrate these points, are as follows:

```
cutv0300
1 1 1 1 7   max. accuracy (prelim,prelim update,final,final update,gradient)
3 5 1 4 7   accurate
5 6 2 6 8   quick/solvent
0 0 0 0 0
0 0 0 0 0
```

The rest of the .cutoff file consists of the cutoff sets. Each set is specified by one line with four integers, sometimes followed by lines containing explicit cutoff keyword values, and ending with a blank line. The four integers represent the variables *jcor* and *kcor* (described below), the grid number, and the number of cutoff values to follow immediately below. The grid number should be 1 for the coarse grid, 2 for the medium grid, 3 for the fine grid, and 4 for the ultrafine grid, 5 for the charge grid, 6 for the gradient grid, 7 for the electron density cubic grid, 8 for the DFT medium grid, or 10 for the DFT gradient grid, where these grids are specified by the keywords **gcoarse**, **gmedium**, **gfine**, **gufine**, **gcharge**, **ggrad**, **geldens**, **gdftmed**, and **gdftgrad**. (The *Grid and Dealiasing Function Keywords* subsection of 8.6 contains more information on these keywords.)

The next lines specify each cutoff by number (e.g. 22 for the variable **cut22**) and value. Thus, the cutoff set:

```
5 2 4 3       set 3
  21 1.0e-3
  22 3.0
  24 1.0e-2
```

means that *jcor* is 5, *kcor* is 2, the ultrafine grid is used, and that three cutoff values which differ from the defaults follow. The next three lines set the cutoff values **cut21**, **cut22**, and **cut24**. (If you need more information on cutoffs, please contact Schrödinger, Inc.)

The variables *jcor* and *kcor* determine what analytic corrections are calculated for a particular SCF iteration. The meanings of their possible values are shown in Table 9.5.1. The variables a, b, and c in

***Table 9.5.1.*** *Determination of calculations of analytic corrections for SCF iterations. (a, b, and c refer to distinct atoms.)*

| variable | value | description |
|---|---|---|
| *jcor* | 0 | no Coulomb terms calculated analytically |
| | 1 | atomic analytic corrections of the form <aa\|aa> calculated for ***J*** |
| | 3 | analytic corrections of the form <aa\|aa> and <aa\|bb> calculated for ***J*** |
| | 4 | analytic corrections of the form <aa\|aa>, <aa\|ab>, <aa\|bb>, and <aa\|bc> calculated for ***J*** |
| | 5 | analytic corrections of the form <aa\|aa>, <aa\|ab>, <aa\|bb>, <ab\|ab>, and <aa\|bc> calculated for ***J*** (diatomic + <aa\|bc>) |
| *kcor* | 0 | no exchange terms calculated analytically |
| | 1 | atomic analytic corrections of the form <aa\|aa> calculated for ***K*** |
| | 2 | diatomic analytic corrections of the form <aa\|aa>, <aa\|ab>, <aa\|bb>, and <ab\|ab> calculated for ***K*** |

the table refer to distinct atoms.

To perform an all-analytic calculation, you can set the keyword **nops** in the **gen** section of the input file to 1. All-analytic calculations use the cutoff keyword values in the .cutoff file, but ignore the *jcor*, *kcor*, and pseudospectral grid information.

# 9.6. THE LEWIS FILE

The Lewis file determines how van der Waals radii for calculations using the Jaguar solvation module are set according to chemical functional groups. By default, for neutral molecules in water, the program calculates a Lewis dot structure for the molecule or system, scans the Lewis file for radius information for each atom and sets radii for relevant atoms, then sets any radii not determined by the Lewis file

according to the **atomic** section or the standard, default value. Settings for radii not included the Lewis file are described in Section 3.5 and section 8.9 and are listed in Table E near the back of the User's Guide. If you do not want the atomic radii that determine the dielectric continuum boundary to change according to the chemical environment of the atom, set the solvation keyword **isurf** to 0 in the **gen** section. Otherwise, Jaguar will alter some radii for neutral molecules by using the default.lewis file from the data directory, unless you specify your own .lewis file in a LEWISFILE line in the input file, as described in section 8.2.

If radii are being set according to a Lewis file, Jaguar first computes a Lewis dot structure for the input geometry to determine each atom's bonds and hybridization type. The element and chemical environment of each atom determine its atom type. When Jaguar reads the Lewis file, it sets the atom's van der Waals radius to the value dictated by the *first* atom type description in the Lewis file that matches that atom. For instance, if the atom were a methyl carbon and the first atom type description in the file was of a carbon bound to a hydrogen, the radius would be set to the radius matching that description, even if a later line in the Lewis file described a carbon bound to three hydrogens.

Atom types are determined by an atom's element and by any combination of the following other properties:

- hybridization (for example, $sp^2$)
- bonding type, which is determined by the bond orders of the bond(s) the atom forms and the element(s) to which the atom is bonded
- hybridization type, which describes the hybridization and element of atoms to which the original atom is bonded
- ring size: the size of the ring the atom is in (for instance, 6 for a carbon in benzene)
- aromaticity of the ring the atom is in, if any (benzene carbons, for instance, form an aromatic ring). An aromatic ring is defined here by the Huckel Rule: if the ring contains 4n + 2 pi electrons, where n is any non-negative integer, it is considered to be aromatic.

The Lewis file first determines the bonding types and hybridization types that will be recognized, then lists atomic radii for various atom types. The file contains different versions of this information for LMP2 calculations than it does for other wavefunction types. Therefore, the first non-blank line of the file should begin

```
CALCULATION TYPE 01
```

(with any comment allowed after this string), indicating that the information following that line is for HF, DFT, or GVB wavefunctions. After all the information in the file for these calculations, the file should contain this line:

```
CALCULATION TYPE 02
```

followed by information for LMP2 wavefunctions.

## *Describing Bonding Types in the Lewis File*

The bonding type information for HF, DFT, or GVB wavefunctions should follow the first line describing the calculation type. The first line of this information should begin

```
BONDING TYPE 01
```

and the rest of the bonding type information should not contain any blank lines except the last line, which signals the end of bonding type information.

Bonding type information should be listed for each relevant element in turn. The information for the first atom should follow immediately after the "BONDING TYPE 01" label. The first character of the information for that atom should begin with the atom's atomic number. The following lines should describe up to five "groups" of bonds for that atom. Each group must begin with the word

```
Group
```

(with no preceding spaces) and must contain information for bonds of bond orders 1, 2, and 3, with a comment line identifying each bond order. The group is simply a list of bonded atoms and bond orders for the element being described—for instance, Group 2 for carbon could describe C=C and C=O bonds by specifying that for bond order 2, Group 2 contains two elements with atom numbers 6 and 8. The first line under each bond order label must list the number of elements for that bond order and that group (2 for the C=C and C=O example); if this number is 0, the next line must list the atomic numbers for those elements (6 and 8 in the example).

Here is the beginning of a sample .lewis file illustrating a list of bonding type information for carbon, including some comments to further explain the file format:

```
CALCULATION TYPE 01  (HF/DFT/GVB)

BONDING TYPE 01 INFORMATION
6   CARBON
Group 1: C-H bonds  (only "Group" must be here; the rest is a comment)
   Bond order: 1  (this should be a non-blank comment line)
     1 element
     1  (the atomic number of H)
   Bond order: 2  (this should be a non-blank comment line)
     0 elements
   Bond order: 3  (this should be a non-blank comment line)
     0 elements
Group 2: C=C and C=O bonds
   Bond order: 1
     0 elements
   Bond order: 2
     2 elements
     6  8
   Bond order: 3
     0 elements
```

The number of spaces at the beginning of the lines described above is irrelevant for all lines except the "Group" lines.

After all the groups have been specified for a particular atom, the file should contain a line containing three asterisks ('***') to indicate the next element's bonding types are about to be described (in the same format). After all desired bonding types are described for all appropriate elements, the bonding type information should end with a blank line.

## *Describing Hybridization Types in the Lewis File*

The hybridization type information in the Lewis file includes up to five groups for each element described, where each group indicates a set of elements and hybridizations for those elements. The hybridization applies to the atom to which the original element is bonded. The information for hydrogen's first group, for instance, could list C (atomic number 6) with $sp^2$ hybridization, allowing a later line in the Lewis file to set a particular radius for hydrogen atoms bonded to $sp^2$ carbons.

The format of the hybridization type information is very similar to that of the bonding type information. The first line of this information (for HF, GVB, or DFT calculations) should begin

```
HYBRIDIZATION TYPE 01
```

and the rest of the hybridization type information should not contain any blank lines except the last line, which signals the end of hybridization type information.

Hybridization type information should be listed for each relevant element in turn. The information for the first atom should follow immediately after the "HYBRIDIZATION TYPE 01" label. The first character of the information for that atom should begin with the atom's atomic number. The following lines should describe up to five hybridization "groups" for that atom. Each group must begin with the word

```
Group
```

(with no preceding spaces). The group is simply a list of bonded atoms for all relevant hybridization types of those bonded atoms—for instance, Group 2 for hydrogen could describe hydrogens bonded to sp carbons by listing carbon's atomic number under an sp hybridization label. Because there is no default number of hybridizations described for each group (unlike for the bonding type information, where each group contained sets for three bond orders), the first line under each group label must begin with the number of hybridizations described for that group (after any number of spaces).

The next line dictates a hybridization for the bonded elements about to be described. Hybridization labels *must start with five spaces*, followed by one of the following character strings:

```
s hybridization
p hybridization
d hybridization
sp hybridization
sp2 hybridization
sp3 hybridization
sp3d hybridization
sp3d2 hybridization
```

For each hybridization, the bonded elements with that hybridization are then listed in two lines, the first indicating the number of elements and the second indicating the elements themselves, as for the bonding type information.

Information for any following atoms should be preceded by a line with three asterisks, and a blank line indicates the end of the hybridization type information, as for the bonding type information.

The beginning of the hybridization information in a sample .lewis file, illustrating a list of hybridization type information for hydrogen and carbon, is shown below, with some comments to further explain the file format:

```
HYBRIDIZATION TYPE 01 INFORMATION
1 HYDROGEN
```

```
Group 1: H-C(sp2) bonds (the Group label must not have spaces before it)
   1 hybridization
     sp2 hybridization ("sp2 hybr..." MUST have 5 spaces before it)
       1 element
       6
Group 2: H-O(sp3) bonds
   1 hybridization
     sp3 hybridization
       1 element
       8
***
6 CARBON
Group 1: C-C(sp3) bonds
   1 hybridization
     sp3 hybridization
       1 atom
       6
```

The number of spaces at the beginning of the lines described above is irrelevant for all lines except the "Group" lines and the hybridization labels.

After all desired hybridization types are described for all appropriate elements, the hybridization type information should end with a blank line.

## *Setting van der Waals Radii from Lewis File Data*

The Lewis file can be used to make non-default choices for van der Waals radii of atoms in particular chemical environments, or even to reset the default radii for particular elements. After Jaguar's lewis program analyzes an input geometry's Lewis dot structure, it sets the atom's van der Waals radius to the value dictated by the *first* atom type description of element and chemical environment in the Lewis file that matches that atom with no contradiction. If no such matching description exists in the Lewis file, the atom is assigned the default radius for that element.

Atom type descriptions in the Lewis file should be preceded by a heading beginning

```
RADII TYPE 01
```

for information applying to HF, GVB, or DFT wavefunctions, or

```
RADII TYPE 02
```

for information for LMP2 wavefunctions. After that, each atom type description is listed. Blank lines are allowed in an atom type description list, and as long as some spacing exists between numbers and

comments on each line, the number of spacing characters is irrelevant. However, keep in mind that the order of the atom type descriptions is important since the first matching description will always be used.

Each line describing an atom type has six integers, one real number, and an optional comment, in that order. The integers describe the atom type, while the real number sets the radius in Angstroms for that atom type. The six integers describe the following characteristics, in turn:

- atomic number (for instance, 6 for carbon)
- hybridization of the atom itself
- bonding type of the atom (elements it is bound to and order of those bonds)
- hybridization type of the atom (hybridization and elements of atoms to which it is bound)
- size of ring (if any) the atom is in
- aromaticity of that ring according to Huckel Rule (aromatic rings have 4n + 2 pi electrons, where n is a non-negative integer)

All six integer values and a corresponding radius value must always be listed in an atom type description line, and the atomic number must correspond to an actual element. However, any or all of the other five integer values can be set to –1, a wild card entry indicating that any value for that characteristic matches that atom type description. To reset a default radius for hydrogen, for instance, you could put the following line before any other descriptions of hydrogen atoms:

```
1  -1  -1  -1  -1  -1   1.10   H all 1.1, ignoring chemical environment
```

and the van der Waals radius for all hydrogen atoms would be set to 1.10 Å.

To describe the hybridization of the atom itself, the atom type description line's second integer should take on one of the values indicated in Table 9.6.1

The description of the atom's bonding type uses the groups listed in the bonding type information described in the subsection *Describing Bonding Types in the Lewis File* above (unless it is –1). Any positive integer for bonding type describes the number of bonds the atom has in each of the bonding type groups for its element and/or the number of all other bonds the atom has. A bonding type group describes

***Table 9.6.1.*** *Lewis file hybridization numbers and corresponding hybridization types.*

| hybridization number | corresponding hybridization type |
|---|---|
| 1 | s hybridization |
| 2 | p hybridization |
| 3 | d hybridization |
| 5 | sp hybridization |
| 6 | $sp^2$ hybridization |
| 7 | $sp^3$ hybridization |
| 8 | $sp^3d$ hybridization |
| 9 | $sp^3d^2$ hybridization |

elements of bonded atoms and orders of those bonds, as described in the subsection *Describing Bonding Types in the Lewis File*. The third integer in an atom type description line determines how many bonds the atom forms of each bonding type group ***g*** for an atom of a particular element, where ***g*** indicates the order of the bonding type groups listed for that element. The number of bonds from group ***g*** is indicated by the $10^g$ digit in the integer.

For example, if ***g*** were 1 and the atom being described were carbon, ***g*** would correspond to the first bonding type group listed for carbon, and a bonding type integer value of 40 ($4 \times 10^1$) would indicate that that carbon atom had four bonds from carbon's Group 1 bonding type information. If the Lewis file contained the bonding type information example provided in the subsection *Describing Bonding Types in the Lewis File*, which included the lines:

```
6   CARBON
Group 1: C-H bonds  (only "Group" must be here; the rest is a comment)
   Bond order: 1  (this should be a non-blank comment line)
     1 element
     1  (the atomic number of H)
```

the integer value of 40 would describe a methane carbon. The same sample Lewis file information, whose key Group 2 information for carbon appears in these lines:

```
Group 2: C=C and C=O bonds
   Bond order: 1
     0 elements
   Bond order: 2
     2 elements
     6  8
```

would mean that this radius information line

```
6  -1  120  -1  -1  -1   2.00   C in H2-C=C or H2-C=O
```

would describe a carbon atom (6) with one bond from carbon's Group 2 (a double bond to either C or O) and two bonds from carbon's Group 1 (single bonds to H), and would set such an atom's radius to 2.00 Å (unless another matching description preceded that line).

The rightmost digit in the integer describing bonding type specifies the number of bonds formed by the atom which are not of any of the forms described in the groups for that atom's bonding type information. A double or triple bond counts as one bond, not two or three, and lone pairs should not be included in the bond count.

The digits of the bonding type integer must describe *all* of an atom's bonding in order to match the atom information. For example, if the Lewis file described above contained no group for C–C bonds in the bonding type information, the integer "200" would only describe a carbon atom with one double bond to another C or O and no other bonds, while the integer "202" would adequately describe a carbon with one double bond to another carbon and two single bonds to other carbon atoms.

The fourth integer in an atom type description, which describes hybridization type, or the elements and hybridization of the atoms to which an atom is bound, works almost the same way as the integer describing bonding type. As it does for bonding types, the digit ***g*** places from the rightmost digit in the integer represents the ***g***th group in the hybridization type information for that element (see the subsection *Describing Hybridization Types in the Lewis File* above for more information), while the rightmost digit specifies the number of bonds to elements and hybridization types that do not fit into any of the groups described for the element of the atom being evaluated. For example, suppose only one hybridization group were described for carbon in the sample Lewis file, as follows:

```
6 CARBON
Group 1: C-C(sp3) bonds
   1 hybridization
     sp3 hybridization
       1 atom
       6
```

Then this atom type description line in a Lewis file would accurately match the middle carbon in methylethylene ($H_2C$=CH–$CH_3$):

```
6  -1   -1  12  -1  -1   2.00   C in H2-C=C or H2-C=O
```

as would the following line, which also contains the proper settings for the middle carbon's hybridization and bonding type:

```
6   6  111  12  -1  -1   2.00   C in H2-C=C or H2-C=O
```

As for the integer describing bonding type, the total of the digits in the fourth integer should be the same as the number of bonds (three for this example, remembering that the double bond counts as one bond)—that is, all bonds should be accounted for (unless, of course, the integer is –1).

The fifth and sixth integers describe the ring the atom is in, if any. If the fifth integer is a positive number ***n***, it indicates that the atom description corresponds to an atom in a ring of size ***n***. (For example, a benzene carbon is in a ring of size 6.) If the fifth number is a negative number –***n***, the description corresponds to an atom in a ring of size ***n*** *or smaller*, unless the fifth integer is –1, in which case the question of the atom's ring environment is ignored completely. The size ***n*** should not be more than 20.

The sixth integer indicates whether the description corresponds to an atom in an aromatic ring as defined by the Huckel Rule (4n + 2 electrons in ring, where n is a non-negative integer). If the sixth integer is 1, the description corresponds to an aromatic ring; if it is 0, the description corresponds to a non-aromatic ring; and if it is –1, the aromaticity of the ring is irrelevant. Note, however, that aromaticity is *not* evaluated if the fifth integer (describing ring size) is –1. To describe aromaticity without regard to ring size, you should generally set the fifth integer to –20 and the sixth to 1, corresponding to atoms in aromatic rings of size 20 or less.

## *Default Behavior for Setting Radii*

The radius settings contained in Jaguar's default.lewis file are used for any relevant atoms in all default solvation calculations in water with Jaguar's solvation module, except for calculations on ions or on

molecules containing atoms with atomic numbers greater than 18. By default, the program uses the first Lewis dot structure generated to evaluate the radii, and the solvation calculation also includes a correction term (the first shell correction factor) that depends on that Lewis dot structure. If the Lewis dot structure does not correspond to that desired for the molecule, the keyword **lewstr** should be changed to correspond to a better structure, as described in the **gen** section description in Chapter 8. To avoid using Lewis dot structures for either correction terms or radius settings, set the **gen** section keyword **isurf** to 0. To use Lewis dot structures to set radii but not for correction terms, **isurf** should be 0 but the keyword **ivanset** should be 1. All Lewis dot keywords are explained in section 8.6 under *GVB and Lewis Dot Structure Keywords*.

The radius settings in the file default.lewis, which appears in the standard data directory, were optimized for HF, GVB, and LMP2 solvation calculations in water with Jaguar's solvation module that included the default correction terms for the cavity and surface area. The molecules used for radius optimization were the molecules containing carbon, hydrogen, oxygen, nitrogen, and sulfur from reference [115]. All calculations used a 6-31G** basis set. Geometries were obtained from gas phase optimizations at the HF, GVB, and LMP2 levels. For both the geometry optimizations and the solvation energy calculations, the GVB and LMP2 treatment was restricted to heteroatom pairs.

# 10. Troubleshooting

Naturally, we hope that you will never need to use this chapter. However, if you have problems using Jaguar, you may find useful advice here. Please also feel free to contact us, as described in Section 1.3.

## 10.1. Problems Getting Started

If you are having problems starting the interface or submitting jobs, please read this section.

Your local system manager should have already installed Jaguar. If the command "jaguar &" does not work because the jaguar command does not exist or if you get an error message regarding installation, contact him or her.

The exact wording of error messages you get when trying to run Jaguar may differ from the error messages described here, depending on your hardware and X implementation. Remember that your X server is either your workstation or the machine that acts as the server for your X terminal, the display host is the workstation or terminal at which you are sitting, and you are trying to start Jaguar as an X client on some machine not necessarily serving as your X server.

Some of the issues addressed here are standard X and/or UNIX issues, and consulting your X and UNIX documentation may help. Also, you may be able to avoid repeatedly entering commands described in this section by including them in your .login, .cshrc, or other startup files in your home directory.

If you can start the interface but you have problems submitting jobs, please skip to the subsections *Problems Related to Your Temporary Directory* and *Problems Running Jaguar Calculations on Other Nodes* below.

## The SCHRODINGER Environment Variable

Before running Jaguar on any particular machine, you must set the environment variable SCHRODINGER to point to the SCHRODINGER directory on that machine. SCHRODINGER is the directory containing Jaguar version 4.2, which is in a subdirectory called "jaguar-vxxxxx," where *xxxxx* is the five-digit version number.

To check whether SCHRODINGER is set, enter the command

```
echo $SCHRODINGER
```

If the output from this command is a directory containing Jaguar—for instance,

```
/scr/jaguar
```

—you can skip the rest of this subsection.

If you determine that SCHRODINGER has not been defined, you will need to set it. If you don't know where the SCHRODINGER directory is, please ask the person who installed Jaguar. Then, if you are running C shell or ksh, enter the command

```
setenv SCHRODINGER directory
```

or if you are using bash or sh, enter the command

```
export SCHRODINGER=directory
```

where for either command, you should substitute the appropriate SCHRODINGER directory for *directory*. You should also set SCHRODINGER within your shell startup file (in the .cshrc file in your home directory if you are running C shell, for instance) by adding the setenv or export command to the file, so that in the future, you won't need to set SCHRODINGER from your terminal window before running Jaguar.

## *Including the jaguar Command in Your Path*

The command "jaguar" is actually a short script that finds the appropriate version of Jaguar to run and passes on any relevant options to the main Jaguar program.

Try entering the command

```
jaguar help
```

If the output from this command was a description of how to use the jaguar command, the person who installed Jaguar has already set things up for you nicely, and you can skip the rest of this subsection. Otherwise, if your machine responded with an error message like

```
jaguar - Command not found
```

you should still be able to run Jaguar by entering the full path name for the jaguar command:

*homedir*/jaguar

where *homedir* is the SCHRODINGER directory described above. However, you will make your future life with Jaguar a lot easier if you either add the directory *homedir* to your path (by editing your shell startup file), or copy the *homedir*/jaguar script into a directory that is in your path.

## *Problems Starting the Interface*

If, when you try to start the Jaguar interface with the command:

```
jaguar &
```

you get an error message that begins:

```
Error: Couldn't find a jaguar.hosts file
```

in your X window, it indicates that Jaguar is unable to locate (or use) the jaguar.hosts file, which provides various setup information. Usually, there should be a jaguar.hosts file in Jaguar's home directory, which is set by the environment variable SCHRODINGER and should be something like /scr/SCHRODINGER. First, make sure you have set SCHRODINGER in your .cshrc or some other profile file. (See the subsection *The SCHRODINGER Environment Variable* above for more information.) Next, from the SCHRODINGER directory, type "ls -l jaguar.hosts" to make sure this file exists and you have

read permission. If you cannot locate the file, or if the permission is set correctly and you still have problems, contact Schrödinger for help.

If you have other problems when you try to start the interface, they are likely to involve permissions needed to do things over a network. Most of these problems never arise if the machines you are using are within a local network. If you are using only local hosts and still get these problems, you might ask your system manager for advice in addition to following the instructions given here.

If you get the message:

```
Error: Can't Open display
```

you are probably trying to start the interface from a machine that is not acting as your X server, and this machine does not know what your display is. Before starting the interface, you may be able to specify the display with the following command, substituting the name of your X server or terminal for "<displayhost>". Type

```
setenv DISPLAY <displayhost>:0.
```

if you use csh, or:

```
export DISPLAY
DISPLAY=<displayhost>:0.
```

if you use sh. Alternatively, you may specify explicitly what display you are using by starting the interface with the command:

```
jaguar -display <displayhost>:0. &
```

An error message of:

```
Xlib:  connection to "<displayhost>:0.0" refused by server
Xlib:  Client is not authorized to connect to Server
Error: Can't Open display
```

usually means one of two things. First, if you are not the person who initially logged onto the X server, you cannot bring up any type of X window on the display. In this case you should log out and log in as yourself. Second, if your X server and the interface host are not the same machine, the X server may not recognize the right of the interface host to display. To correct this problem, type the following in a window on your X server:

```
xauth nextract - <displayhost>:0.0 | rsh <ihost> xauth nmerge -
```

Here, <ihost> should be replaced by the name of the interface host. Also, the "remote shell" command should be used for "rsh"; usually /bin/ucb/rsh serves this purpose, but "rsh" gives /usr/bin/remsh on some machines. If the "restricted shell" rsh precedes the remote shell one in your path, you must use the full path name. If the xauth command listed above results in "xauth: Command not found.," your path probably does not include /usr/bin/X11, and you should substitute "/usr/bin/X11/xauth" for "xauth" in the command and try again. If the xauth command yields "Permission denied," the rsh command was not allowed, and you should make sure to read the paragraphs on rsh and rcp commands in the subsection *Problems Running Jaguar Calculations on Other Nodes*.

If you have problems running the xauth command described in the above paragraph, an alternative is to simply type:

```
xhost +<ihost>
```

on your X server. This command allows anyone (including yourself) logged onto <ihost> to run X programs on <displayhost>. Since this command is a potential security risk, it is not recommended as a permanent solution.

If you are using an SGI and you get an error like this:

```
dgl error (getservbyname): unknown service sgi-dgl/tcp
```

it means Jaguar is unable to find the SGI Distributed Graphics Library. The file /etc/services should contain this information in a line beginning "sgi-dgl". If this line is commented out (that is, if it begins with a "#" character) you can try uncommenting it. If you continue to have this problem and it is affecting the interface performance—for example, if you are unable to display molecules on your SGI as described in Section 2.5—you should probably ask your system administrator for help.

## *Problems Related to Your Temporary Directory*

When you run a Jaguar job, Jaguar generates various files it needs during the calculation within a temporary directory (often within a directory called /scr, /tmp, or something similar). At the end of the job, the program deletes most files in this directory by default, copying back only the output file and any other files you requested. If you get an error related to temp directory space when you try to run Jaguar, the program is probably having trouble getting access to the temp directory space it needs to run.

If you are using the interface to run jobs, you can tell what temporary space Jaguar will try to use by looking at the Temp directory setting in the Run window. The program actually makes a subdirectory named after the job within this directory and writes files there. For instance, if a person with the user name erwin had a Temp directory listing of /scr/erwin for a job called h2o (with an input file called h2o.in), Jaguar would attempt to create the directory /scr/erwin/h2o and write files there during the job.

If your job gives error messages related to the temp directory, you should check to make sure that the temp directory listed in the Run window exists and that you have write permission within that directory. For example, if the output

```
Error creating or cd-ing to temp directory:
   /scr/erwin/h2o
```

appeared in the h2o.log file for erwin's job, it would probably be because /scr/erwin did not exist or because erwin did not have permission to make the subdirectory h2o within it.

You may need someone to create the appropriate temp directory or change permissions on it from the root account. Use the command "ls -l" to get information on ownership of your temp directory or the directory above it. If you need to be able to create a subdirectory within a directory owned by root or another account that does not belong to you and for which you do not have write permission, please contact your system administrator for help.

## *Problems Running Jaguar Calculations on Other Nodes*

In order for the interface to launch jobs on other nodes and for these nodes to copy files back to the interface host, they must be able to do rsh (remote shell) jobs on each other and rcp (remote copy) files between them. If you get a "Permission denied" error when trying to start a job (by selecting OK in the Run window, as described in Section 2.1 and Section 2.7), the rsh command is not being allowed. This problem may occur even if the interface and the host where the calculation is to be performed are the same. The best method to test whether this problem is occurring is to issue individual rsh commands at an interface host command line prompt, such as

```
rsh calculation-host who
```

where you substitute the name of the host where you want to perform the calculation for *calculation-host*.

If both the interface and calculation hosts are on the same local network, ask your system manager about allowing rsh commands between the two, which could be done in several ways, depending on your system. One way is to list hosts which are allowed to rsh to a given host in its /etc/hosts.equiv file. It may be necessary to include the interface host's own name in its /etc/hosts.equiv file if the calculation is to be done on the interface host. See your system manager or your UNIX documentation concerning trusted hosts, NIS domains, or networking for more information.

If you get an error which refers to problems writing or cd-ing to a temp directory for the job, you should make sure that you have permission to write to the directory specified in the Temp directory bar in the Run window, and that you have permission to create that directory if it does not already exist.

If you are unable to allow rsh's machine-wide as described above (e.g., your interface and calculation hosts are not local to each other), you must include the interface machine in a .rhosts file in your home directory on the calculation host, and vice versa. If you have the same user name on both nodes, a line in the .rhosts file only needs to contain the entire host name. For more information, see the man page on rhosts on your machine.

One further complication can result if you have distinct user names on the interface and calculation hosts. In this case, you may get an error like one of the following:

```
Login incorrect
remshd: Login incorrect
rshd: xxxx-xxxx  The remote user login is not correct.
```

This problem generally occurs only when the interface and calculation hosts are on separate local area networks. To handle these distinct sites, you must use a personal jaguar.hosts file. Each host line in the file should include your user name on that host in the following format:

```
host:   sgi schrod@anny.schrodinger.com
```

where the name of the machine in the "host:" field matches that in the "uname -n" command output for that machine, where "schrod" is replaced by your user name, and "anny.schrodinger.com" is replaced by the name of your calculation host. See section 6.1 for details on how to construct your own jaguar.hosts file.

# 10.2. ON-LINE HELP

You can obtain help from any window you use by pressing the Help button found within it. Pressing Help brings up the Help window with the appropriate topic already selected.

You may also find it convenient to select Topic... from under the Help heading, which brings up the Help window. You can see on-line help on a variety of subjects by clicking on them as they are listed under the Help items heading so that they show up in the Selection bar, then hitting Select, or by rapidly double-clicking on the topic name.

# 10.3. OTHER PROBLEMS

Some other problems you may encounter are detailed below, along with solutions or explanations.

- *You cannot read in a particular file as input.* Make sure you are specifying the right file type in the bar under the directory and file name lists. The File type bar selection alters both the extensions for the file names listed (which can be changed by editing the Filter bar), but also determines the format the interface expects the file to have. Also, make sure the file name, and not just its directory, is really showing up in the Selection bar before you hit OK. See Section 2.4 for more information.

- *The molecular structure for the calculation is not what you expected it to be.* If you read in a Jaguar input, .bgf, .hes, .dat, or GAUSSIAN input file, or if you read in a file containing only a geometry, the geometry will be obtained from that file, unless you edit the geometry after reading the file. Any geometry you entered before reading the file will be wiped out. Also, if you symmetrize the geometry, as described in Section , or set symmetry on for the calculation, as described in Section , Jaguar may make small changes to the molecular geometry. If these changes are a problem, you should avoid symmetrizing the geometry and possibly turn the symmetry option off as well.

- *The calculation is not what you expected it to be.* If you read in a Jaguar input file or a GAUSSIAN input file, some of the settings in the file will take precedence over settings previously made in the interface. See Section 2.4 for more details. Also, certain settings affect other settings automatically—for instance, if you choose to

calculate polarizabilities, the energy convergence criterion can reset to $1.0 \times 10^{-6}$.

- *For a GVB job, the program exits early and the output states you need a different number of lone pairs on a particular atom.* As described under *GVB or GVB-RCI Pair Input* in Section 3.3 and under *GVB Calculations: GVB Pair Selection* in section 6.3, you must specify lone pairs for either all or none of the lone pairs on any particular atom. Change the lone pair information and try running the calculation again.

- *The SCF calculation does not converge properly, or frequencies or other properties look wrong.* If the geometry entered is of poor quality, the calculation may not converge properly, which may also lead to inaccurate calculation of molecular properties. If you are performing a geometry optimization, check to see whether the geometry changes are reasonable; if you are performing a single-point calculation, make sure the structure entered is appropriate. You may wish to minimize the structure with a molecular mechanics program before inputting it. If the structure is reasonable, convergence problems should not occur, and we would appreciate it if you would describe them to us at the address given in Section 1.3, preferably by e-mailing us the input, output, and log files for the job with a brief explanation. To get converged results in the meantime, you can try using level shifting and/or setting the accuracy level to ultrafine, both of which are described in Section 3.9 (under *Convergence Issues* and *Accuracy Level*). The calculation will be slower, but convergence may be better for problem cases.

- *The settings available in the* Read, Save, *or* Run *windows are not what you expected them to be.* Many of the options for these windows are determined by the jaguar.hosts file used for the job. This file is the jaguar.hosts file found in the directory from which the interface was started, if it exists; otherwise, it is the jaguar.hosts file in your home directory, if that file exists; and if neither of those two files exists, the default jaguar.hosts file for the interface host is used. (You can find out which jaguar.hosts file is being used by clicking on About in the top row of buttons in the interface's main window, then clicking on Schrödinger and looking at the configuration file listed.) If you are using a different jaguar.hosts file than you expect, or if you are working with a new version of Jaguar and a new jaguar.hosts file has been installed on your system, you should examine the jaguar.hosts file for the job and make sure it is in the same form as the file for the system for the version of Jaguar that you are using, and that the

settings are appropriate. See section 6.1 for a description of the jaguar.hosts file.

- *The job fails with a memory-related error ("Memory fault," "out of memory," or "no memory available for array," for example) or the log file indicates "Killed" for the job.* Your job may have failed because the machine was too heavily loaded, in which case rerunning the job when the load is lower could solve the problem. Otherwise, you may want to try an appropriate setting from the *Memory Usage Keywords* subsection of section 8.6 to avoid a problem for a large job, or you (and/or your system manager) may want to investigate increasing the maximum virtual size or the "soft" limit allowed for memory on your machine. (For HP workstations in particular, the default memory limits are quite small.) Please contact us, as described in Section 1.3, if you would like any tips for setting memory use for your machine.

# 11. The Parallel Jaguar Module

## 11.1. About Parallel Jaguar

The parallel module for Jaguar is sold and licensed separately. Jaguar can run in SMP (symmetric multi-processing) mode on shared-memory architectures (as in workstations that contain multiple processors) or on distributed-memory architectures (as in IBM SP2 clusters or Linux Beowulf clusters). Jaguar can also run on clusters in which each node contains multiple processors. Parallel Jaguar is not multi-threaded.

## 11.2. Installing Parallel Jaguar

Parallel Jaguar is currently available for the SGI, Linux, IBM, HP, Compaq, and Sun platforms. The parallel Jaguar executables are installed by default when you install Jaguar. After installation, you will need to edit the file $SCHRODINGER/jaguar.hosts and add an 'nprocs' line to the entry for each computer indicating how many cpus are available on that computer. For computer clusters, the value of 'nprocs' for *each node* listed in the jaguar.hosts file should be the total number of processors available in the cluster. See Section 6.1 for details of the format for the jaguar.hosts file.

To actually run parallel Jaguar, you will need a Jaguar license code that includes permission to run the parallel module. If you have purchased parallel Jaguar or are evaluating it, but you don't already have such a license code, please send the output from the command

"jaguar machid" to **help@schrodinger.com** and we will send you an appropriate license line.

# 11.3. Requirements for Different Computer Platforms

For all platforms, be sure to use local disks for scratch space. Performance will be significantly reduced if an NFS-mounted scratch disk is used. Also, avoid using scratch directories which are actually symbolic links. Using symbolic links for scratch directories is known to prevent Jaguar jobs from running, especially under Linux, although it might be shell-dependent. Thus, if /scratch is actually a symbolic link to /scr, specify /scr in the jaguar.hosts file rather than /scratch.

## *SGI*

There are two system requirements for SGI: the Message-Passing Toolkit (MPT) v1.2.1.1 or later, and Array Services 3.1 or later. If you are using the PBS batch queue system, you will need MPT v1.5.0.0 or later. These packages must be installed by the system administrator for your computer because it will require root permission. Following are installation instructions:

1. Check to see if the required MPI (Message Passing Interface) files are already installed with the command

```
showprods | grep MPI
```

If MPI is not installed, you can install it from the MPT package, which can be downloaded from

http://www.sgi.com/products/evaluation/

2. If necessary, install Array Services 3.1 or later. You can check to see if Array Services are present with the command

```
showprods | grep arraysvcs
```

If you have Array 3.2, Patch 3532 is required. You can check if Patch 3532 is already installed with the command

```
showprods | grep 3532
```

If it is not, you can get it from

http://support.sgi.com/surfzone/patches/

Array services allows your SGI to run MPI applications like parallel Jaguar. Start the array services daemon with the following command:

```
/etc/init.d/array start
```

The arrayd daemon can be configured to start automatically at system startup with the command

```
chkconfig array on
```

Finally, note that a CERT advisory (http://www.cert.org/advisories/CA-99-09-arrayd.html) has been posted about the default Array Services installation on IRIX versions 6.2 - 6.5.4. A simple fix is available at

ftp://patches.sgi.com/support/free/security/advisories/19990701-01-P

## *LINUX*

For Linux, parallel Jaguar requires the MPICH package. We also recommend Linux kernel 2.2.17 or any later 2.2 kernel. Jaguar may not function correctly with earlier 2.2 kernels. It will probably work with Linux 2.4 kernels, but this has not been extensively tested. If Jaguar is to run in parallel on a multiprocessor machine, the kernel must be compiled for SMP (symmetric multiprocessing).

***Installing MPICH*** Parallel Jaguar for Linux has only been qualified for MPICH versions 1.2.0 through 1.2.2.

To check whether you already have MPICH installed, execute the command

```
which mpirun
```

If the 'mpirun' script is not found, try

```
rpm -qi mpich
```

If this is successful, then MPICH is installed but the executables are not yet in your PATH. Verify the location of the MPICH executables by saying

```
rpm -ql mpich | grep mpirun
```

and make sure the directory containing the 'mpirun' script is added to your PATH, as described below.

**Binary installation** A precompiled binary RedHat 6.2 package for MPICH 1.2.0 is available from the Schrödinger ftp site at the following location:

ftp://ftp.schrodinger.com/support/mpich-1.2.0-14.i386.rpm

We recommend that you use this particular rpm package if at all possible. Other rpm packages of MPICH, which may not be built under RedHat 6.2, can be found at the RPMFind website, rpmfind.net (just do a search for 'mpich'). We do not recommend the use of of the MPICH package with the version number 1.2.0-5, because one of the utility scripts supplied with it is broken.

To install the rpm you must have system administrator (root) privileges. After downloading the .rpm file to your machine(s), install it with the following command:

```
rpm -Uvh mpich-1.2.0-14.i386.rpm
```

on each machine where parallel Jaguar will be run. This will install the mpich executables and support files under /usr/lib/mpich, and documentation (including html files) in /usr/share/doc/mpich-1.2.0. The *MPICH User's Guide* is /usr/share/doc/mpich-1.2.0/guide.ps.gz; you may find it useful to refer to this for more detailed instructions on configuring and using MPICH.

**Source installation** If you want to compile a current version of MPICH yourself, the latest sources are always available from

http://www-unix.mcs.anl.gov/mpi/mpich/

Instructions for building and installing MPICH are included with the source code.

***Configuration***

1) Add the MPICH bin directory to the PATH environment variable. This is necessary in order for Jaguar to find the 'mpirun' launch script. If you installed the rpm of MPICH, the bin directory is /usr/lib/mpich/bin. Csh (or tcsh) users should add the following line to their ~/.cshrc file: (or to ~/.tcshrc for tcsh users)

```
setenv PATH /usr/lib/mpich/bin:$PATH
```

Bash users should add the line

```
export PATH=/usr/lib/mpich/bin:$PATH
```

to their ~/.bashrc file.

2) Edit the file /usr/lib/mpich/share/machines.LINUX and list the names of the computers which Jaguar may use for parallel calculations. Each line of the machines.LINUX file should specify the name of a computer and the number of processors on that computer, separated by a colon. The computer name should match the output of the 'hostname' command. For example:

```
homer.mynet.edu:2
marge.mynet.edu:2
bart.mynet.edu:1
```

3) Edit the jaguar.hosts file in the directory where Jaguar was installed, and list in it the names of the computers in the machines.LINUX file. The computer names in jaguar.hosts need not include the domain name. See Section 6.1 for details on the format of the jaguar.hosts file. For each computer, the value used for 'nprocs' should be the *total* number of processors for all machines in the machines.LINUX file. A jaguar.hosts file corresponding to the machines.LINUX file in the above example would look like:

```
host:        homer
home:        /apps/Schrodinger
temproot:    /scr
nprocs:      5
!
host:        marge
home:        /apps/Schrodinger
temproot:    /scr
nprocs:      5
!
host:        bart
home:        /apps/Schrodinger
temproot:    /scr
nprocs:      5
```

4) Launch the secure server. The processes in a parallel Jaguar job communicate using the MPICH secure server, which is called *serv_p4*. The secure server must be running on all computers on which Jaguar is to run in parallel, which is normally all machines listed in the machines.LINUX file. Rather than login to each node and launch the server, one can use the chp4_servs script in /usr/lib/mpich/sbin to do this automatically. The launch command must include the number of a communications port which the secure server can use. The port number should be a four-digit number greater than 1023. If the port number chosen is 1234, then the command to launch the *serv_p4* server on all nodes would be:

```
usr/lib/mpich/sbin/chp4_servs -port=1234
```

If the above command does not work for some reason, the secure server will have to be launched manually on each machine with the following command:

```
usr/lib/mpich/bin/serv_p4 -o -p 1234
```

You can verify that the server really is running on each node using the script /usr/lib/mpich/sbin/chkserv. This is a perl5 script and may require some modification.

Although each user may choose a port number for his/her own use, we recommend that the system administrator launch the server as root so that all users can use the same port number. Each user must also set the following environment variables:

csh and tcsh:

```
setenv MPI_USEP4SSPORT yes
setenv MPI_P4SSPORT 1234
```

bash:

```
export MPI_USEP4SSPORT=yes
export MPI_P4SSPORT=1234
```

Note that the port number assigned to MPI_P4SSPORT must match the port number used to launch the *serv_p4* server. If all users use the same port number, then they can just set these environment variables in their login scripts to avoid having to set them manually at each session.

5) Ensure that rsh is enabled. By default, Jaguar uses rsh to communicate with remote nodes (even if you are running on a stand-alone SMP workstation with 2 cpus, rsh will still be used). To enable rsh, each user must create a file called '.rhosts' in his or her home directory. The .rhosts file should contain the name of each computer listed in the machines.LINUX file, followed by the user's login name, e.g.,

```
homer.mynet.edu <username>
marge.mynet.edu <username>
bart.mynet.edu <username>
```

The .rhosts file must be owned by the user (not by root) and must not be writable by anyone except the user, or authentication will fail. To ensure this, use the command

```
chmod 644 ~/.rhosts
```

We strongly recommend that you test rsh connections by using the 'tstmachines' shell script, which is in /usr/lib/mpich/sbin. This script will launch several rsh commands to all of the computers listed in the machines.LINUX file, and it will let you know if there are any problems. If the command is successful it will simply return the prompt with no output. You can also use the '-v' flag on the command line to see exactly what the script is doing.

## *IBM*

For IBM, you need to install the Parallel Operating Environment (POE) package, which includes the MPI libraries. Jaguar requires POE v2.4 or later. Be sure to check the README file in /usr/lpp/ppe.poe/ and the manpage for details on POE. If you use LoadLeveler, it must be v2.1 or later.

You will need to set an environment variable in order to use multiple processors for a job. The variable you need to set depends on how your machine has been configured; specifically whether you are running the Job Manager or not. The Job Manager manages pools of nodes, and assigns specific parallel jobs to specific nodes. To test whether you are using the Job Manager, type

```
ps aux | grep jmd
```

If you see 'jmd' processes listed, you are running the Job Manager. In this case, you need to tell Job Manager the pool from which you wish to have nodes assigned to you. The command 'jm_status -P' will list the available pools and their member nodes. The environment variable that sets your job pool is called MP_RMPOOL, and it should be set to the appropriate pool number:

```
export MP_RMPOOL=1 (for ksh or bash)
setenv MP_RMPOOL 1 (for csh)
```

If your machine does not use the Job Manager, then the environment variable that you need to set is called MP_HOSTFILE. This environment variable should point to a hostfile that contains the name(s) of the nodes on which parallel jobs can be run. The node name is listed once for each processor in that node. For example, suppose you have a workstation called 'bobcat' with 4 processors in it and you want to be able to use up to all 4 processors for a parallel job. The hostfile would contain the following four lines:

```
bobcat
bobcat
bobcat
bobcat
```

If you call this hostfile 'my.hostfile', then you would set MP_HOSTFILE as follows:

```
export MP_HOSTFILE=/home/mbeachy/my.hostfile
                                (for ksh or bash)

setenv MP_HOSTFILE /home/mbeachy/my.hostfile
                                      (for csh)
```

If you are unsure of your system configuration, please contact your system administrator for more information.

Ensure that the jaguar.hosts file is properly configured for your cluster. In particular, the 'nprocs' setting for each node should be equal to the total number of processors in the cluster. See Section 6.1.

Finally, you must also make sure you have rsh access to the machine, even if you are on it. To do this, add a line to your ~/.rhosts file that specifies the node that you need access to, and your login name:

bobcat.schrodinger.com mbeachy

This gives user 'mbeachy' rsh access to machine 'bobcat.schrodinger.com'.

When you run a parallel Jaguar job on an IBM, you will currently see ERROR messages printed to the log file after the completion of each executable:

```
ERROR: 0031-636  User requested or EOF termination of
pm_command
```

This is not really an error, so please ignore these messages. There is no way for us to turn them off at this time. We will do our best to remove them in the future.

## *HP, Compaq, Sun*

To run Jaguar in parallel on HP, Compaq, and Sun computers, you just need to install MPI. For HP, it should be MPI v1.05 or later. For Compaq, it should be v1.9 or later. On Suns, it should be v3.1 or later.

If a parallel Jaguar job is killed on a Compaq computer, or if Jaguar dies in such a way that it can't perform any cleanup, the operating system may not free up the shared memory segments and semaphores associated with the job. We suggest running the 'ipcs' command from time to time to check for shared memory segments and semaphores that have not been freed up. If you find some, run the MPI utility 'mpiclean' to free them. Note that mpiclean will not work unless all processes associated with the failed job have been terminated.

# 11.4. RUNNING JOBS IN PARALLEL

Once parallel Jaguar is installed, running Jaguar jobs in parallel is easy. You need only specify the number of processors to use for the job at the time you launch it. If you launch a Jaguar job from the command line, you use the "-p" flag to indicate the number of processors to be used:

```
jaguar run -p 8 <jobname>
```

or

```
jaguar batch -p 8 <list of jobnames>
```

If you launch the job from the graphical interface, just type the number of processors to be used into the "# of Processors:" window in the Jaguar Run panel. By default, the maximum number of processors that you can request is shown in this window. The number here is read from the "nprocs:" line in the jaguar.hosts file.

An alternative to the "-p" command line option is to set the environment variable JAGUAR_NPROCS to a particular value (e.g. 8). If you do that, your Jaguar jobs will run in parallel on JAGUAR_NPROCS processors.

If you need to pass additional parameters to the POE or MPI launch commands, you can do this with the environment variables JAGUAR_POE_FLAGS (for IBMs) or JAGUAR_MPI_FLAGS (for all other platforms). Simply set the appropriate environment variable to the argument(s) that you want to pass in. For example, if you want verbose output from mpirun on an SGI, set the following:

```
setenv JAGUAR_MPI_FLAGS "-v"
```

RULE OF THUMB FOR NUMBER OF PROCESSORS: When you are choosing the number of processors to use for a parallel Jaguar job,

take the number of basis functions for the job and divide by 100 (for HF or DFT jobs) or 80 (for LMP2 jobs), then discard any portion of this number after the decimal place. This number is the maximum number of processors advised for an efficient parallel run. For instance, if your molecule had 486 basis functions, the maximum number of processors advised for an HF or DFT calculation would be 4, and the maximum number of processors for an LMP2 job would be 6.

You can tell whether a job is running in parallel by looking at its log file (jobname.log). If the job is running in parallel, the third line of the log file will say, for example,

```
Running on   2 processors
```

If there is no such line, the job is running in serial mode.

If you are using a queue system for your parallel jobs, you may need to take into account that the number of processes created by Jaguar is generally the number of processors for the job plus one (because the Jaguar control program jexec always runs as a separate process).

The following kinds of jobs can be run in parallel:

- HF and DFT single-point calculations (in gas phase or in solution)
- HF and DFT geometry optimizations (in gas phase or in solution)
- closed-shell LMP2 single-point calculations

If you want to compute analytic frequencies, you probably should run your Jaguar energy calculation or geometry optimization first in parallel, then use the restart job for a frequency calculation in serial mode.

Jobs that cannot be run in parallel mode include

- jobs that use all-analytic SCF methods
- LMP2 geometry optimization, frequency, solvation, polarizability, and/or multipole jobs
- open-shell LMP2 jobs
- GVB, GVB-RCI, and GVB-LMP2 jobs
- CPHF (hyper)polarizability jobs
- LMP2 jobs with more processors than LMP2 orbitals

- jobs with more processors than atoms

We hope to add parallel capabilities for some of these features in the future.

The development of parallel Jaguar is discussed in references [117] and [118].

If you have any questions or comments concerning parallel Jaguar, please contact Schrödinger, Inc. by e-mail at **help@schrodinger.com**, or by phone at (503) 299-1150.

# 12. The $pK_a$ Prediction Module

## 12.1. INTRODUCTION

Schrödinger's $pK_a$ prediction module represents the first attempt to utilize ab initio quantum chemical methods to reliably predict $pK_a$'s in aqueous media. The module employs a combination of correlated ab initio quantum chemistry, a self-consistent reaction field (SCRF) continuum treatment of solvation, and empirical corrections to repair deficiencies in both the ab initio and continuum solvation models. This combination leads to high accuracy for a wide range of organic compounds, in conjunction with tractable computational requirements.

The user interface to the methodology has been designed to avoid the necessity of running the many individual jobs required to assemble the various components of the calculation. Schrödinger has optimized each of the components for the best tradeoffs of accuracy versus efficiency. The empirical correction terms, which have been developed for the centrally important ionizable groups relevant to the chemical and pharmaceutical industries, are specifically designed to work with the basis sets, electron correlation levels, and solvation model of the ab initio methodology. The transferability of the corrections has been tested by examining a sizeable set of test molecules, as will be documented below. Subsequent releases of the module will incorporate additional functional groups and more sophisticated treatment of

conformationally flexible molecules, and will provide increasingly improved accuracy.

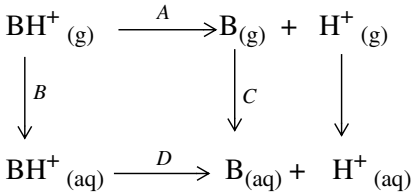
Several features of the method distinguish it from purely empirical fragment-based approaches, which are complementary to the present product. Firstly, we expect that the use of ab initio quantum chemistry rather than fragment table lookups and interpolation will lead to a substantially wider range of applicability, as well as significantly higher precision when the compound in question is not a direct entry in the empirical table. Secondly, our methods allow for a reasonable treatment of conformational effects, which are in general entirely missing from fragment-based methods. Optimal use of the methodology in this fashion is accomplished by performing solution phase conformational searches via the Macromodel molecular modeling code. Thirdly, the method can handle multiple protonation states in a systematic fashion.

This chapter is divided into four subsequent sections. First, the basic theory of $pK_a$ calculation is explicated, including a discussion of the empirical correction approach. Then, a discussion of key issues in using the program in complex situations (conformational flexibility, multiple protonation states) is given. Thirdly, results from our internal suite of test cases are presented. Finally, a practical tutorial describing how to set up, run, and interpret jobs is presented.

# 12.2. THEORY OF $pK_A$ CALCULATION

## *Ab initio Quantum Chemical Calculation of $pK_a$'s*

The calculation of the $pK_a$ of a molecule in aqueous solution can be represented as a thermodynamic cycle:

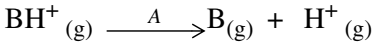



The strategy in our $pK_a$ module is to calculate parts *A, B*, and *C* of the above cycle, whereupon the actual $pK_a$, which is related to *D* via

$$pK_a = \frac{1}{2.3\,RT} D$$

can be obtained by summing these three components (experimental value of –259.5 kcal/mol is used for the solvation free energy change of a proton).

Segment A is the gas phase reaction:




$\Delta$ The gas phase free energy difference between the protonated and deprotonated states can be computed via the usual relations:

$$A = D\,H - T\,D\,S$$

$$= E_{B(g)} - E_{BH+(g)} + 5/2\ RT - TDS$$

Evaluation of this expression requires the following quantum chemical calculations:

(1) *Geometry optimization* of the protonated and deprotonated species. Note that quantum chemical methods generally carry out a conjugate gradient optimization and hence cannot search for multiple minima. We shall assume in this discussion that there is only a single well-defined conformational minimum and that a good initial guess, obtained, for example, from molecular mechanics or semiempirical quantum chemistry, is available. Density functional theory, particularly those variants employing an admixture of Hartree-Fock exchange, have been shown to provide good quality geometries; we utilize B3LYP/6-31G* geometry optimization.

(2) *Accurate single point energies* at each optimized geometry must be evaluated. These single point calculations are carried out at a significantly higher level of theory than the geometry optimization; however, since only one energy is required, the overall cost of this step is in fact less than that for geometry optimization. In recent publications, and in our own extensive unpublished work, the B3LYP method with large basis sets has been shown to yield excellent gas phase energetics for deprotonation reactions, with errors typically in the 1-3 kcal/mole range. We employ the cc-pVTZ(+) basis set of Dunning and coworkers in the present methodology. cc-pVTZ(+) represents a mixed basis set where cc-pVTZ+ is used for atoms involved in the deprotonation reaction, while cc-pVTZ covers the rest. The residual errors in the DFT calculations appear to be relatively constant for a given functional group as the substituents are altered, and hence can be ameliorated considerably by the empirical corrections.

(3) *The solvation free energy* of the protonated and deprotonated species must be computed. We have chosen to do this using the gas phase geometries, an approximation that we have tested and shown to be sufficient for the present purposes (some of the errors induced are compensated by the empirical parameterization).

As we have discussed extensively in several publications, empirical optimization of parameters is absolutely necessary to obtain accurate solvation free energies from SCRF calculation, no matter what the level of electron correlation. Continuum solvation methods do not

rigorously treat effects at the dielectric boundary, which therefore must be adjusted to fit experiment.

For neutral species, we have optimized parameters (both dielectric radii and surface tension terms) by fitting to experimental gas to water solvation free energy data for small molecules. Agreement to within a few tenths of a kcal/mole can be obtained for most functional groups. However, parameterization of the model for ionic species in this fashion cannot lead to high levels of accuracy, because there are large error bars on the experimental data (typically 5-10 kcal/mole). An error of 5 kcal/mole in the solvation free energy, which was not systematic, would lead to huge errors in $pK_a$ calculations. This is because in determining the $pK_a$ there is a cancellation of two very large terms: the gas phase deprotonation energy (which favors the protonated state) and the solvation free energy (which favors the deprotonated state). Errors in either term therefore can be a small percentage of the total energy but lead to very large errors in the resulting calculated $pK_a$.

To overcome this problem, we have adopted a novel strategy, which is to fit the parameters for ions *directly to experimental $pK_a$ data.* If the gas phase quantum chemistry and neutral solvation are reliably computed, then the solvation free energy of the ionic species becomes the remaining unknown quantity. Since $pK_a$ measurements are carried out to quite high precision (in contrast to direct measurements of ionic solvation), fitting to this data does not lead to the large uncertainties that would be associated with the ionic solvation data. Additionally, there is an exceptionally large database of known $pK_a$ values for a wide range of chemical functional groups.

In general, the dielectric radii of ions (particularly negative ions) are expected to be smaller than that for the corresponding neutral species, due to the phenomenon of electrostriction. In our fitting procedure, the ionic radii are adjusted to yield the smoothest and most consistent results for the members of the training set for each functional group. For anions, special radii are assigned to the principal location of the negative charge; for cations, radii are assigned to hydrogens on the proton acceptor and to the proton acceptor itself. Functional groups for which radii have been developed are tabulated in section IV below. For novel functional groups with divergent electronic properties, reparameterization of the model to a subset of experimental data is advisable, as the results are rather sensitive to these quantities. However, the current model is able to robustly handle substituent and conformational effects once a functional group is parameterized.

In our work on neutral solvation, we have found that it is necessary to supplement parameterization of dielectric radii with surface area terms to correct for first shell hydrogen bonding; a purely electrostatic model is incapable, by itself, of properly describing such interactions for all molecules. For ions, these terms are expected to be even larger and more important, as the magnitude of the first shell hydrogen bonding interactions are 3-5 times larger than in neutrals. However, what we have done in the present model is to incorporate these corrections into our overall empirical fitting scheme, described below. In this fashion, all of the errors associated with the various components of the method are subsumed into a small number of parameters characteristic of the functional group in question.

## *Empirical Corrections*

The results of the above calculation can be assembled to yield a raw $pK_a$ value. Because of the intrinsic errors involved in each step, it is necessary to apply an empirical correction scheme to the raw data to yield good agreement with experiment. The validity of this scheme can be assessed only by comparison with experimental data. For the most important functional groups, we have examined a large and diverse set of molecules (including those containing polyfunctional groups and conformational flexibility) to evaluate the robustness of the methodology. For the molecules considered below, it appears to be quite satisfactory. For example, for protonation of nitrogens in heterocycles, an average prediction accuracy of 0.19 is obtained over 16 molecules whose $pK_a$'s range from 0.65 to 9.17.

Our empirical corrections take the simple linear form:

$$pK_a = A\ pK_a\ (raw) +$$

That is, we assume that the correction terms obey a linear free energy relationship. The B term is similar to our previously employed surface tension corrections for neutral solvation. The linear term takes into account the significant variation in charge on the ionizable group as a function of substituents. Consider, for example, carboxylic acids. The charge on the oxygens in the $CO_2^-$ moiety varies by as much as 0.45 eu as one goes from electron withdrawing substituents (oxalic acid) to electron donating substituents (propionic acid). This change in charge will alter the appropriate hydrogen bonding first shell correction term as well as the solvation free energy computed by the SCRF calculation. Since changes in the raw $pK_a$ are well correlated

with these charge shifts, linearly scaling the correction term to the raw $pK_a$ is capable of capturing this effect.

While corrections to the solvation model are the dominant terms in our empirical corrections, there are also intrinsic errors in the gas phase DFT calculations, which are implicitly incorporated into the correction scheme. The assumption is that these errors are systematic for a given functional group. This means that the DFT calculations are required only to reproduce the relative energetic changes produced by modification of substituents, a less demanding task than absolute $pK_a$ prediction. As we are aiming at high accuracy (0.5 $pK_a$ units or better) which is somewhat beyond the raw capabilities of the DFT calculations, the corrections are again necessary for success in this endeavor.

# 12.3. PREDICTING PKA'S IN COMPLEX SYSTEMS

## *Overview*

The algorithm described in section II can be straightforwardly applied in the simplest case, characterized as follows:

(1) There is only one relevant ionizable group in the molecule.

(2) There is a single relevant conformation of the molecule and this conformation is valid for both the protonated and deprotonated form.

An example of this situation would be acetic acid.

However, it is also possible to use the module in more complex situations. In what follows, we explain how this is accomplished.

## *Conformational Flexibility*

Let us first consider the case in which assumption (1) above holds, but the protonated and deprotonated states can each exist in multiple conformations, which may be energetically competitive. There are several possible ways in which the conformational problem can be addressed. In Version 1.0 of the module (the current release), only

method (1) below has been automated. It is still possible to carry out the strategies outlined in method (2), however at present the user must run multiple jobs and manually assemble the data into a final result.

(1) The user can submit one protonated and one deprotonated conformation, which are assumed to dominate the phase space due to being lowest in energy in their respective class. This is probably a reasonable assumption for many problems. Note that *the conformation that is lowest in the protonated state many not be lowest in the deprotonated state.* There are in many cases obvious electrostatic reasons why a conformational change upon protonation/deprotonation would occur. The program is set up to accept a different conformation for each species.

The selection of the appropriate conformation can be nontrivial. Our recommendation is to do a solution phase conformational search in Macromodel, using the MMFF force field and the GB/SA continuum solvent model. This is a very fast procedure and gives a reasonable ordering of conformational free energies in solution. Alternatively, one could either construct the conformation by hand or employ a gas phase conformational searching protocol. Preliminary results indicate that there may be situations where a solution phase conformational search is necessary to obtain accurate results.

(2) A more accurate approach would be to carry out quantum chemical calculations for multiple conformations (again generated from a Macromodel solution phase conformational search) and use all of this information to compute the $pK_a$. Two ways of doing this are;

- Pick the lowest solution phase free energy conformer for each protonation state and compute the $pK_a$ from this. This is analogous to (1) above but allows for imprecision in the conformational search protocol. It is also obviously more demanding in terms of CPU time.
- One can carry out a statistical mechanical average over conformations to determine the "average" $pK_a$. The assumption made if this option is chosen is that the midpoint of the $pK_a$ titration curve is achieved when the total population of the deprotonated state, summing over all deprotonated conformations, is equal to the total population of the protonated state, also summing over all conformations. This approach should be more accurate than that described in (a), although how important statistical effects are in practice remains to be ascertained.

## *Equivalent Sites*

Some molecules have two or more equivalent sites for protonation or deprotonation. Examples include ethanediamine, the analogous dicarboxylic acid, or the molecule melamine in our suite of test cases, which has three equivalent sites. In this situation, there is a statistical correction factor arising from increased entropy of the appropriate species. As we do not have an automated facility for recognizing equivalent sites in the current version of the program, the user must make this correction by hand to the result obtained from running the $pK_a$ prediction module. The correction factor is $\log(N^2)$, where $N$ is the number of equivalent sites, and the power of 2 comes from the fact that there are two particles involved: $H^+$ and the species being protonated. For convenience, we supply here the correction factors for two and three equivalent sites (room temperature):

Bases:

- 2 equivalent protonation sites: +0.60
- 3 equivalent protonation sites: +0.95

Acids:

- 2 equivalent deprotonation sites: -0.60
- 3 equivalent deprotonation sites: -0.95

## *Multiple Protonation Sites*

Many molecules have several sites, which can have different $pK_a$'s. Let us consider here a case with two distinct possible protonation sites, and suppose we want to calculate the $pK_a$ of site 1. Then the following situations are possible:

(1) The two $pK_a$'s are well separated and the $pK_a$ of site 1 is higher than that of site 2. In this case, site 2 will be deprotonated when site 1 is being titrated in an experiment. Therefore, one should run a site 1 $pK_a$ calculation with site 2 in the deprotonated state.

(2) The two $pK_a$'s are well separated and the $pK_a$ of site 2 is higher than that of site 1. In this case, site 2 will be protonated when site 1 is being titrated in an experiment. Therefore, one should run a site 1 $pK_a$ calculation with site 2 in the protonated state.

(3) The two $pK_a$'s are unknown, or the $pK_a$'s are close together. In this case, there are a total of four protonation states to run: both sites

protonated, one site protonated (two cases), and no sites protonated. If one obtains data for these four cases, the titration curve can be assembled and one can make comparison with experiment.

Cases (1) and (2) are straightforward to handle using the current version. When the groups are close together, there are some scientific issues about the accuracy of results for multiply protonated states. This will become clearer as more test cases are run. In the next release, we intend to treat case (3) inside the program. At present, though, the users must run two separate $pK_a$ jobs (each of which will handle two of the four protonation states) and build the titration curve themselves.

# 12.4. RESULTS

Table I presents a summary of the results for the functional groups that we have parameterized, including the number of cases studied, average deviation from experiment, and maximum deviation. A listing of the results for the individual test cases, comparing experimental and calculated $pK_a$'s can be found in Table II.

The largest set of test cases examined have been for carboxylic acids and nitrogen bases in heterocyclic rings. The latter cases have minimal conformational flexibility and hence should be easier to handle, and this is indeed reflected in the remarkably low average error of 0.3 and maximum error of 0.9 that we observe.

The carboxylic acids include some examples with polyfunctional groups and significant flexibility. We have not carried out an exhaustive analysis of the conformational energetics for these cases; hence much of the deviation from experiment that we report may be due to this. Nevertheless, the errors are quite respectable.

The largest error for alcohols comes from t-butanol, with the predicted $pK_a$ being too low (acidic) by about 3 $pK_a$ units. We believe the source of this error to come from the overstabilization of the ionized form, $(CH_3)_3CO^-$. It is possible that the continuum solvation model does not fully account for the steric shielding from the three methyl groups on the negatively charged oxygen.

For functional groups where a relatively small number of compounds have been included in the parameterization, the results are obviously less reliable. We have nevertheless included some groups of this type

(hydroxamic acids, sulfinic acids, sulfonic acids, thiophenol and imine) in the initial release. Feedback in these cases as to the validity of the parameterization would be particularly valuable to us in developing the next generation of parameters.

***Table 12.4.1.*** *Functional groups for which pKa parameters are available.*

| **FUNCTIONAL GROUP** | **Mean Abs. Dev.** | **Max. Abs. Dev.** | **.jres file group number** |
|---|---|---|---|
| ALCOHOLS | 0.8 | 3.0 | 1 |
| PHENOLS | 0.1 | 0.3 | 2 |
| CARBOXYLIC ACIDS | 0.5 | 1.9 | 4 |
| THIOLS | 0.2 | 0.3 | 10 |
| SULFONAMIDES | 0.7 | 2.3 | 15 |
| HYDROXAMIC ACIDS | 0.4 | 1.4 | 6 |
| IMIDES | 0.8 | 1.5 | 13 |
| BARBITURIC ACIDS | 0.1 | 0.2 | 12 |
| TETRAZOLES | 0.6 | 1.5 | 17 |
| PRIMARY AMINES | 0.4 | 0.6 | 31 |
| SECONDARY AMINES | 0.3 | 1.0 | 32 |
| TERTIARY AMINES | 0.6 | 1.4 | 33 |
| ANILINES | 0.2 | 0.5 | 25-30 |
| HETEROCYCLES | 0.2 | 0.6 | 19 |
| AMIDINES | 0.5 | 0.9 | 22 |
| BENZODIAZEPINES | 0.4 | 0.5 | 23 |
| GUANIDINES | 0.6 | 1.3 | 21 |
| PYRROLES (C-2 protonation) | 0.4 | 0.6 | 35 |
| INDOLES (C-3 protonation) | 0.1 | 0.3 | 36 |
| **Total Mean Abs. Dev.** | **0.4** | | |

***Table 12.4.2.*** *Molecules used in the pKa parameterization, arranged by functional group*

| MOLECULE | pKa calc. | pKa exp. | Dev. |
|---|---|---|---|
| **ALCOHOLS** | | | |
| methanol | 16.4 | 15.5 | 0.9 |
| ethanol | 15.9 | 15.9 | 0.0 |
| propanol | 16.0 | 16.2 | -0.2 |
| i-propanol | 15.9 | 17.1 | -1.2 |
| 2-butanol | 16.9 | 17.6 | -0.7 |
| t-butanol | 16.2 | 19.2 | -3.0 |
| allylalcohol | 15.3 | 15.5 | -0.2 |
| propargylalcohol | 15.0 | 14.3 | 0.7 |
| 2-chloroethanol | 13.8 | 12.9 | 0.9 |
| 2,2-dichloroethanol | 12.6 | 12.2 | 0.3 |
| 2,2,2-trichloroethanol | 11.6 | 12.4 | -0.8 |
| 2,2,2-trifluoroethanol | 15.6 | 15.4 | 0.2 |
| 1,2-ethanediol | 13.4 | 13.6 | -0.1 |
| 1,2-propanediol | 15.4 | 14.9 | 0.5 |
| 1,3-propanediol | 16.4 | 15.1 | 1.3 |
| 1,4-butanediol | 16.4 | 15.1 | 1.3 |
| | | | |
| **PHENOLS** | | | |
| phenol | 9.8 | 10.0 | -0.2 |
| 4-aminophenol | 9.3 | 9.4 | -0.1 |
| 4-chlorophnol | 9.6 | 9.9 | -0.3 |
| 4-fluorophenol | 10.4 | 10.2 | 0.2 |
| 4-methoxyphenol | 10.3 | 10.3 | 0.0 |
| 4-methylphenol | 10.6 | 10.5 | 0.2 |
| 4-nitrophenol | 7.3 | 7.2 | 0.2 |
| p-xylol | 10.4 | 10.3 | 0.0 |
| 4-hydroxybenzaldehyde | 7.6 | 7.6 | 0.0 |

| MOLECULE | pKa calc. | pKa exp. | Dev. |
|---|---|---|---|
| **CARBOXYLIC ACIDS** | | | |
| cis 1,2-cyclopropanedicarboxylic | 4.3 | 3.6 | 0.7 |
| trans 1,2-cyclopropanedicarboxylic | 3.9 | 3.8 | 0.1 |
| cis 2-chlorobut-2-enecarboxylic | 3.3 | 2.8 | 0.5 |
| trans 2-chlorobut-2-enecarboxylic | 3.0 | 3.2 | -0.2 |
| 2-chlorobut-3-enecarboxylic | 2.7 | 2.5 | 0.2 |
| 2-chloropropanecarboxylic | 3.0 | 2.9 | 0.1 |
| 2,2-dimethylpropanoic | 4.3 | 5.0 | -0.8 |
| 2-furanecarboxylic | 3.3 | 3.2 | 0.1 |
| cis 2-methylcyclopropanecarboxylic | 4.1 | 5.0 | -0.9 |
| trans 2-methylcyclopropanecarboxylic | 4.4 | 5.0 | -0.6 |
| 2-methylpropanecarboxylic | 4.5 | 4.6 | -0.1 |
| cis 3-chlorobut-2-enecarboxylic | 3.9 | 4.1 | -0.2 |
| rans 3-chlorobut-2-enecarboxylic | 3.5 | 3.9 | -0.5 |
| 3-chloropropanecarboxylic | 4.3 | 4.1 | 0.2 |
| cis 3-chloropropenecaroxylic | 3.9 | 3.5 | 0.4 |
| trans 3-chloropropenecarboxylic | 3.6 | 3.8 | -0.2 |
| 3-chloropropynecarboxylic | 2.9 | 1.9 | 1.0 |
| 3-nitro-2-propanecarboxylic | 4.5 | 2.6 | 1.9 |
| 3-oxopropanecarboxylic | 5.3 | 3.6 | 1.7 |
| cis 4-chlorobut-3-enecarboxylic | 4.4 | 4.1 | 0.3 |
| trans 4-chlorobut-3-enecarboxylic | 3.9 | 4.1 | -0.2 |
| acetic acid | 3.7 | 4.8 | -1.1 |
| acrylic acid | 3.8 | 4.2 | -0.5 |
| benzoic acid | 3.9 | 4.2 | -0.3 |
| butanoic acid | 4.2 | 4.8 | -0.6 |
| trans cinnamic acid | 4.3 | 4.4 | -0.1 |
| formic acid | 2.9 | 3.8 | -0.8 |
| glycolic acid | 3.4 | 3.8 | -0.5 |
| glyoxylic acid | 1.6 | 2.3 | -0.7 |

| MOLECULE | pKa calc. | pKa exp. | Dev. |
|---|---|---|---|
| **CARBOXYLIC ACIDS cont.** | | | |
| malic acid | 2.7 | 3.5 | -0.8 |
| malonic acid | 3.4 | 2.9 | 0.6 |
| oxalic acid | 2.0 | 1.2 | 0.8 |
| pentafluoropropanoic acid | 0.5 | -0.4 | 0.9 |
| propanoic acid | 4.1 | 4.9 | -0.8 |
| propargylic acid | 2.7 | 1.9 | 0.8 |
| succinic acid | 4.1 | 4.2 | -0.2 |
| dl tartaric acid | 3.2 | 3.0 | 0.2 |
| meso tartaric acid | 2.4 | 3.2 | -0.8 |
| tartonic acid | 2.4 | 2.4 | 0.0 |
| trifluoroacetic acid | 0.4 | 0.2 | 0.2 |
| | | | |
| **THIOLS** | | | |
| methylthiol | 10.0 | 10.3 | -0.3 |
| ethylthiol | 10.8 | 10.6 | 0.2 |
| 2-mercaptoethanol | 9.4 | 9.4 | 0.0 |
| 1,2-ethanedithiol | 9.2 | 9.1 | 0.2 |
| | | | |
| **SULFONAMIDES** | | | |
| N-chlorotolylsulfonamide | 4.3 | 4.5 | -0.2 |
| dichlorphenamide | 6.5 | 7.4 | -0.9 |
| mafenide | 9.4 | 8.5 | 0.9 |
| methanesulfonamide | 10.1 | 10.5 | -0.4 |
| nimesulide | 6.3 | 5.9 | 0.4 |
| quinethazone | 9.1 | 9.3 | -0.2 |
| saccharin | 3.0 | 1.6 | 1.4 |
| sulfamethizole | 3.2 | 5.4 | -2.3 |
| sulfaperin | 7.2 | 6.8 | 0.5 |
| sulfacetamide | 5.6 | 5.4 | 0.2 |

| MOLECULE | pKa calc. | pKa exp. | Dev. |
|---|---|---|---|
| **SULFONAMIDES cont.** | | | |
| sulfadiazine | 7.0 | 6.5 | 0.5 |
| sulfadimethoxine | 7.2 | 6.0 | 1.2 |
| sulfamethazine | 7.7 | 7.4 | 0.3 |
| sulfanylamide | 10.4 | 10.4 | -0.1 |
| sulfapyridine | 7.8 | 8.4 | -0.6 |
| sulfaquinoxaline | 6.4 | 5.5 | 0.9 |
| sulthiame | 9.1 | 10.0 | -0.9 |
| xipamide | 9.3 | 10.0 | -0.7 |
| | | | |
| **HYDROXAMIC ACIDS** | | | |
| formohydroxamic | 8.0 | 8.7 | -0.6 |
| acetohydroxamic | 8.5 | 8.7 | -0.2 |
| benzohydroxamic | 8.5 | 8.8 | -0.3 |
| salicylhydroxamic | 8.4 | 7.5 | 1.0 |
| 2-aminobenzohydroxamic | 9.0 | 9.0 | 0.0 |
| 2-chlorobenzohydroxamic | 8.3 | 7.8 | 0.5 |
| 2-fluorobenzohydroxamic | 8.2 | 8.0 | 0.2 |
| 2-nitrobenzohydroxamic | 8.5 | 7.0 | 1.4 |
| 3-nitrobenzohydroxamic | 8.2 | 8.4 | -0.2 |
| 4-aminobenzohydroxamic | 8.8 | 9.4 | -0.6 |
| 4-chlorobenzohydroxamic | 8.4 | 8.7 | -0.3 |
| 4-fluorobenzohydroxamic | 8.4 | 8.8 | -0.4 |
| 4-nitrobenzohydroxamic | 8.2 | 8.3 | -0.1 |
| 4-hydroxybenzohydroxamic | 8.6 | 8.9 | -0.3 |
| | | | |
| **IMIDES** | | | |
| fluorouracil | 8.6 | 8.0 | 0.6 |
| methylthiouracil | 7.9 | 8.2 | -0.3 |
| phenytoin | 8.0 | 8.3 | -0.3 |

| MOLECULE | pKa calc. | pKa exp. | Dev. |
|---|---|---|---|
| **IMIDES cont.** | | | |
| 3,3-methylphenylglutarimide | 10.2 | 9.2 | 1.0 |
| 3,3-dimethylsuccinimid | 8.9 | 9.5 | -0.6 |
| dimethadione | 7.6 | 6.1 | 1.5 |
| phtaleimid | 8.8 | 9.9 | -1.1 |
| succinimid | 8.7 | 9.6 | -0.9 |
| | | | |
| **BARBITURIC ACIDS** | | | |
| 5,5-methylphenylbarbituric | 7.5 | 7.4 | 0.1 |
| 1,5,5-trimethylbarbituric | 8.3 | 8.3 | 0.0 |
| hexobarbital | 8.2 | 8.2 | 0.0 |
| 5,5-dimethylbarbituric | 8.1 | 8.0 | 0.1 |
| 1,5-dimethyl-5-phenylbarbituric | 7.6 | 7.8 | -0.2 |
| | | | |
| **TETRAZOLES** | | | |
| 5-cyclopropyltetrazole | 4.9 | 5.4 | -0.5 |
| 5-methyltetrazole | 4.8 | 5.6 | -0.8 |
| 5-hydroxytetrazole | 5.0 | 5.4 | -0.4 |
| 5-phenoxytetrazole | 4.6 | 4.4 | 0.2 |
| 5-phenyltetrazole | 5.0 | 3.5 | 1.5 |
| tetrazole | 4.8 | 4.9 | -0.1 |
| | | | |
| **PRIMARY AMINES** | | | |
| methylamine | 10.5 | 10.2 | 0.3 |
| ethylamine | 11.0 | 10.6 | 0.3 |
| propylamine | 10.7 | 10.6 | 0.1 |
| t-butylamine | 10.5 | 10.7 | -0.2 |
| 2-aminoethanol | 9.8 | 9.2 | 0.6 |
| 1,2-ethanediamine | 10.1 | 10.7 | -0.6 |
| 1,3-propanediamine | 10.4 | 10.9 | -0.5 |

| MOLECULE | pKa calc. | pKa exp. | Dev. |
|---|---|---|---|
| **SECONDARY AMINES** | | | |
| dimethylamine | 10.9 | 10.7 | 0.2 |
| diethylamine | 11.1 | 11.0 | 0.0 |
| azetidine | 11.3 | 11.3 | 0.0 |
| pyrrolidine | 11.1 | 11.3 | -0.1 |
| piperidine | 11.1 | 11.1 | 0.0 |
| morpholine | 9.5 | 8.5 | 1.0 |
| 2,5-diazahexane | 9.4 | 10.4 | -1.0 |
| | | | |
| **TERTIARY AMINES** | | | |
| trimethylamine | 10.1 | 9.8 | 0.3 |
| triethylamine | 10.6 | 11.0 | -0.4 |
| tripropylamine | 9.2 | 10.7 | -1.4 |
| 1-methylpiperidine | 10.4 | 10.2 | 0.2 |
| triallylamine | 7.1 | 8.3 | -1.3 |
| 1-allylpiperidine | 9.9 | 9.7 | 0.3 |
| dimethylcyclohexylamine | 10.6 | 10.7 | -0.1 |
| dimethylbenzylamine | 8.9 | 9.0 | -0.1 |
| diethylbenzylamine | 9.2 | 9.5 | -0.2 |
| hexamethylenetetramine | 6.5 | 5.3 | 1.3 |
| DABCO | 9.6 | 8.2 | 1.4 |
| | | | |
| **ANILINES** | | | |
| aniline | 4.7 | 4.6 | 0.1 |
| 4-chloroaniline | 4.0 | 4.0 | 0.1 |
| 4-methoxyaniline | 5.5 | 5.2 | 0.3 |
| 4-nitroaniline | 1.1 | 1.0 | 0.1 |
| p-toluidine | 4.6 | 5.1 | -0.5 |
| **AMIDINES** | | | |
| imidazo[2,3-b]thioxazole | 8.1 | 8.0 | 0.1 |

| MOLECULE | pKa calc. | pKa exp. | Dev. |
|---|---|---|---|
| **AMIDINES cont.** | | | |
| tetrahydrozoline | 9.6 | 10.5 | -0.9 |
| hydroxyimidazo[2,3-a]isoindole | 9.1 | 8.6 | 0.5 |
| tolazoline | 10.6 | 10.3 | 0.3 |
| | | | |
| **HETEROCYCLES** | | | |
| 2-aminopyridine | 7.2 | 6.7 | 0.5 |
| 2-aminothiazole | 5.5 | 5.4 | 0.2 |
| 2-methylimidazole | 7.9 | 8.0 | -0.1 |
| 3-aminopyridine | 6.1 | 6.0 | 0.0 |
| 4-aminopyridine | 9.6 | 9.7 | -0.1 |
| 4-methylpyridine | 6.2 | 6.0 | 0.2 |
| benzimidazole | 5.2 | 5.8 | -0.6 |
| imidazole | 6.8 | 7.0 | -0.2 |
| isoquinoline | 5.4 | 5.4 | 0.1 |
| melamine | 5.1 | 5.0 | 0.1 |
| pyrazine | 1.0 | 0.7 | 0.4 |
| pyrazole | 2.5 | 2.5 | 0.1 |
| pyridine | 5.2 | 5.3 | -0.1 |
| pyrimidine | 1.1 | 1.3 | -0.2 |
| quinoline | 5.0 | 4.8 | 0.1 |
| thiazole | 2.4 | 2.8 | -0.4 |
| | | | |
| **BENZODIAZEPINES** | | | |
| 1,3-dihydro-1-methyl-5-phenyl-1,4-benzodiazepin-2-one | 3.8 | 3.3 | 0.5 |
| 1,3-dihydro-3-hydroxy-5-phenyl-1,4-benzodiazepin-2one | 1.9 | 1.7 | 0.2 |
| 1,3-dihydro-3-hydroxy-1-methyl-5-phenyl-1,4-benzodiazepin-2one | 1.4 | 1.6 | -0.2 |
| 1,3-dihydro-5-phenyl-1,4-benzodiazepin-2one | 4.0 | 3.5 | 0.5 |

| MOLECULE | pKa calc. | pKa exp. | Dev. |
|---|---|---|---|
| **BENZODIAZEPINES cont.** | | | |
| 2,3-dihydro-1-methyl-5-phenyl-1,4-benzodiazepine | 6.1 | 6.2 | -0.1 |
| 3-hydro-2-methylamine-4-oxy-5-phenyl-1,4-benzodiazepine | 3.9 | 4.8 | -0.9 |
| | | | |
| **GUANIDINES** | | | |
| clonidine | 8.2 | 8.1 | 0.1 |
| debrisoquin | 13.0 | 11.9 | 1.1 |
| guanidine | 12.5 | 13.8 | -1.3 |
| methylguanidine | 13.4 | 13.4 | 0.0 |
| | | | |
| **PYRROLES** (C-2 protonation) | | | |
| pyrrole | -4.1 | -3.8 | -0.3 |
| 1-methylpyrrole | -2.3 | -2.9 | 0.6 |
| 2-methylpyrrole | -0.7 | -0.2 | -0.5 |
| 3-methylpyrrole | -0.9 | -1.0 | 0.1 |
| | | | |
| **INDOLES** (C-3 protonation) | | | |
| indole | -3.7 | -3.6 | -0.1 |
| 1-methylindole | -2.0 | -2.3 | 0.3 |
| 2-methylindole | -0.4 | -0.3 | -0.1 |
| 3-methylindole | -4.6 | -4.6 | 0.0 |

# 12.5. GUIDE TO RUNNING THE PROGRAM

## *Installing the pKa Module*

To run the pKa module, you will need:

- Jaguar version 3.5, release 37 or later.
- A Jaguar license (in the file $JAGUAR_HOME/license for versions prior to v4.1 and $SCHRODINGER/license for v4.1 or later) that lets you run the $pK_a$ module.

To install the pKa module, simply follow the instructions in the *Schrödinger Product Installation Guide*. After you have successfully installed release 37 (or later), you should send in the machid information to obtain a license activating the pKa module – please explicitly indicate in your license request that you are going to run $pK_a$ calculations.

## *JAGUAR Input Files for $pK_a$ Calculations*

$pK_a$ calculations require input files in Jaguar format containing a molecular geometry and a labeled acidic site. The acidic site is either an acidic hydrogen in acids, or a heteroatom to be protonated in bases. If the starting geometry is not a neutral molecule but an ion, you have to specify its formal charge in the &atomic section (see section 8.9 in the Jaguar manual). Also, if the geometry files are not in Jaguar format, you can translate them using the Read and Save windows of the Jaguar GUI (see sections 2.4 and 2.7 in the Jaguar manual) or Babel (type "jaguar babel" for usage instructions).

The acidic site can be marked:

1. by adding the suffix "_pk" after the atomic symbol, or

2. by setting the &gen section keyword "ipkat" to either the atom's name, or to the atom's order number in the zmat section.

Here are three equivalent input file examples for formic acid:

```
1.
&zmat
  C1   1.0590559100      0.0794463600      0.3608319800
  O2   0.8609619100      1.1054614700     -0.2390046100
  O3   2.2130316700     -0.6129886300      0.3489813100
  H_pk 2.8258867600     -0.1221771000     -0.2269021000
  H2   0.3281776900     -0.4358328800      1.0011835800
&



2.(a)
&zmat
  C1   1.0590559100      0.0794463600      0.3608319800
  O2   0.8609619100      1.1054614700     -0.2390046100
  O3   2.2130316700     -0.6129886300      0.3489813100
  H1   2.8258867600     -0.1221771000     -0.2269021000
  H2   0.3281776900     -0.4358328800      1.0011835800
&
&gen
ipkat=H1
&

2.(b)
&zmat
  C1   1.0590559100      0.0794463600      0.3608319800
  O2   0.8609619100      1.1054614700     -0.2390046100
  O3   2.2130316700     -0.6129886300      0.3489813100
  H1   2.8258867600     -0.1221771000     -0.2269021000
  H2   0.3281776900     -0.4358328800      1.0011835800
&
&gen
ipkat=4
&
```

## *Running $pK_a$ Calculations*

To submit a $pK_a$ job using the graphical user interface, follow the instructions in section 2.7 of the Jaguar manual for running batch jobs.

Choose pka.bat as the batch file, and select your $pK_a$ Jaguar input files as the input files for the batch job.

To submit a $pK_a$ job using a command line, type:

```
jaguar batch pka jobname1 [ jobname2 ... ]
```

where pka.bat is the batch file for $pK_a$ jobs, and jobname1.in, jobname2.in, etc. are Jaguar input files for $pK_a$ jobs in the format described above. Use of the wildcard in job names is allowed.

## *Monitoring $pK_a$ Calculations*

The $pK_a$ calculations can be monitored from the interface's Check Job window (if you ran it from the interface) or by looking at the file pka.*.blog (where * is a process identification number).

For each molecule Jaguar creates a jobname_pka subdirectory in the local directory and writes the output from 12 calculations there.

Special suffixes explain what is calculated in each step:

Conjugate acid (either the acid molecule "as is", or protonated base)

| | |
|---|---|
| dft_h | B3LYP/6-31G* geometry optimization |
| nrg_h | B3LYP/cc-pVTZ(-f)(+) single point energy |
| solv_h | B3LYP/6-31G**(+) single point solution phase calculation |
| pr#_h | input file preparation runs |

Conjugate base (deprotonated acid or base "as is")

| | |
|---|---|
| dft | B3LYP/6-31G* geometry optimization |
| nrg | B3LYP/cc-pVTZ(-f)(+) single point energy |
| solv | B3LYP/6-31G**(+) single point solution phase calculation |
| pr# | input file preparation runs |

Final $pK_a$ and $pK_b$ values are calculated from data in these output files and written in file(s) jobname.out in the local directory (where jobname.in is a Jaguar input file submitted for a $pK_a$ calculation). For example, here is the final output file for formic acid:

| Stoichiometry | Charge | pK |
|---|---|---|
| CH2O2 | 0 | pKa : 3.2 |
| CH2O | -1 | pKb : 10.8 |

To list in a table all the $pK_a$ (and/or $pK_b$) values, you can use the 'jaguar results' feature,

```
jaguar results -title -jobname -pka -pkb *.out
```

## *Initial Geometry*

It is very important to choose the lowest energy conformer for the $pK_a$ calculations. As ab initio geometry optimizers only find the local minima, for flexible systems (long n-alkyl chains, many rotatable bonds) we strongly suggest first running a conformational search to

determine the global minimum energy structure, which can then be used as an initial structure for the $pK_a$ run.

For certain systems it is expected that the protonated and deprotonated structure assume different conformations in their lowest energy state. For such cases, create Jaguar input files for both of them, add the suffix "_2input" to the name of the conjugate base input file (jobname_2input.in), and use pka_2input.bat instead of pka.bat to run $pK_a$ calculations. Output files and results will be in the same format as in the single initial geometry runs.

# Information by Element

The tables on the following pages provide information about how each atomic element can be treated within Jaguar. The first four tables describe the atoms and basis sets for which pseudospectral calculations can be performed. The next three tables list atomic radii used for various purposes. Please read each table caption carefully when accessing information in these tables.

Only elements which can be treated with Jaguar are listed in these periodic tables.

# Pseudospectral 3-21G, 6-31G, and 6-311G Availability

***Table A.*** *Current availability of 3-21G, 6-31G, and 6-311G (6-311) basis sets for pseudospectral calculations. The + and ++ diffuse function options are available for any pseudospectral calculations using the 6-31G basis set. Polarization functions are available for the 3-21G basis set when it is listed as 3-21G*, and ** after a basis set's name indicates availability of * and ** options for that set.*

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1 H**<br>3-21G<br>6-31G**<br>6-311** | | | | | | | | | | | | | | | | | **2 He**<br>3-21G<br>6-31G** |
| **3 Li**<br>3-21G<br>6-31G**<br>6-311** | **4 Be**<br>3-21G<br>6-31G** | | | | | | | | | | | | **5 B**<br>3-21G<br>6-31G** | **6 C**<br>3-21G<br>6-31G**<br>6-311** | **7 N**<br>3-21G<br>6-31G**<br>6-311** | **8 O**<br>3-21G<br>6-31G**<br>6-311** | **9 F**<br>3-21G<br>6-31G**<br>6-311** | **10 Ne**<br>3-21G<br>6-31G** |
| **11 Na**<br>3-21G*<br>6-31G**<br>6-311** | **12 Mg**<br>3-21G*<br>6-31G** | | | | | | | | | | | | **13 Al**<br>3-21G*<br>6-31G** | **14 Si**<br>3-21G*<br>6-31G**<br>6-311** | **15 P**<br>3-21G*<br>6-31G**<br>6-311** | **16 S**<br>3-21G*<br>6-31G**<br>6-311** | **17 Cl**<br>3-21G*<br>6-31G**<br>6-311** | **18 Ar**<br>3-21G*<br>6-31G** |
| **19 K** | **20 Ca** | **21 Sc** | **22 Ti** | **23 V** | **24 Cr** | **25 Mn** | **26 Fe** | **27 Co** | **28 Ni** | **29 Cu** | **30 Zn** | **31 Ga** | **32 Ge** | **33 As** | **34 Se** | **35 Br** | **36 Kr** |
| **37 Rb** | **38 Sr** | **39 Y** | **40 Zr** | **41 Nb** | **42 Mo** | **43 Tc** | **44 Ru** | **45 Rh** | **46 Pd** | **47 Ag** | **48 Cd** | **49 In** | **50 Sn** | **51 Sb** | **52 Te** | **53 I** | **54 Xe** |
| **55 Cs** | **56 Ba** | **57 La** | **72 Hf** | **73 Ta** | **74 W** | **75 Re** | **76 Os** | **77 Ir** | **78 Pt** | **79 Au** | **80 Hg** | **81 Tl** | **82 Pb** | **83 Bi** | | | |

# Pseudospectral D95, cc-pVDZ, and cc-pVTZ Availability

***Table B.*** *Current availability of D95, cc-pVDZ, and cc-pVTZ basis sets for pseudospectral calculations. For the D95 basis set, the polarization function options * and ** are available for pseudospectral calculations with any atoms listed below. The + and ++ diffuse function options are available for pseudospectral cc-pVTZ calculations with any atoms listed below. The cc-pVTZ basis set in Jaguar does not yet include f functions.*

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1 H**<br>D95**<br>cc-pVDZ<br>cc-pVTZ | | | | | | | | | | | | | | | | | **2 He** |
| **3 Li**<br>D95** | **4 Be** | | | | | | | | | | | **5 B** | **6 C**<br>D95**<br>cc-pVDZ<br>cc-pVTZ | **7 N**<br>D95**<br>cc-pVDZ<br>cc-pVTZ | **8 O**<br>D95**<br>cc-pVDZ<br>cc-pVTZ | **9 F**<br>D95**<br>cc-pVDZ<br>cc-pVTZ | **10 Ne** |
| **11 Na** | **12 Mg** | | | | | | | | | | | **13 Al** | **14 Si**<br>D95**<br>cc-pVDZ<br>cc-pVTZ | **15 P**<br>D95**<br>cc-pVDZ<br>cc-pVTZ | **16 S**<br>D95**<br>cc-pVDZ<br>cc-pVTZ | **17 Cl**<br>D95**<br>cc-pVDZ<br>cc-pVTZ | **18 Ar** |
| **19 K** | **20 Ca** | **21 Sc** | **22 Ti** | **23 V** | **24 Cr** | **25 Mn** | **26 Fe** | **27 Co** | **28 Ni** | **29 Cu** | **30 Zn** | **31 Ga** | **32 Ge** | **33 As** | **34 Se** | **35 Br** | **36 Kr** |
| **37 Rb** | **38 Sr** | **39 Y** | **40 Zr** | **41 Nb** | **42 Mo** | **43 Tc** | **44 Ru** | **45 Rh** | **46 Pd** | **47 Ag** | **48 Cd** | **49 In** | **50 Sn** | **51 Sb** | **52 Te** | **53 I** | **54 Xe** |
| **55 Cs** | **56 Ba** | **57 La** | **72 Hf** | **73 Ta** | **74 W** | **75 Re** | **76 Os** | **77 Ir** | **78 Pt** | **79 Au** | **80 Hg** | **81 Tl** | **82 Pb** | **83 Bi** | | | |

# Treatment of Atoms for Pseudospectral LAV3P Calculations

***Table C.*** *Calculations that use the LAV3P basis set use LAV3P effective core potentials for some atoms and 6-31G basis functions (with or without * or ** polarization function options) for other atoms For instance, a calculation on $Al_2O_3$ using the LAV3P** basis set would employ 6-31G** basis functions on the oxygen atoms and LAV3P effective core potentials for the Al atoms. Any calculation with the LAV3P basis set (with or without polarization functions) can be performed pseudospectrally.*

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1 H** 6-31G** | | | | | | | | | | | | | | | | | **2 He** 6-31G** |
| **3 Li** 6-31G** | **4 Be** 6-31G** | | | | | | | | | | | **5 B** 6-31G** | **6 C** 6-31G** | **7 N** 6-31G** | **8 O** 6-31G** | **9 F** 6-31G** | **10 Ne** 6-31G** |
| **11 Na** LAV3P | **12 Mg** LAV3P | | | | | | | | | | | **13 Al** LAV3P | **14 Si** LAV3P | **15 P** LAV3P | **16 S** LAV3P | **17 Cl** LAV3P | **18 Ar** LAV3P |
| **19 K** LAV3P | **20 Ca** LAV3P | **21 Sc** LAV3P | **22 Ti** LAV3P | **23 V** LAV3P | **24 Cr** LAV3P | **25 Mn** LAV3P | **26 Fe** LAV3P | **27 Co** LAV3P | **28 Ni** LAV3P | **29 Cu** LAV3P | **30 Zn** LAV3P | **31 Ga** LAV3P | **32 Ge** LAV3P | **33 As** LAV3P | **34 Se** LAV3P | **35 Br** LAV3P | **36 Kr** LAV3P |
| **37 Rb** LAV3P | **38 Sr** LAV3P | **39 Y** LAV3P | **40 Zr** LAV3P | **41 Nb** LAV3P | **42 Mo** LAV3P | **43 Tc** LAV3P | **44 Ru** LAV3P | **45 Rh** LAV3P | **46 Pd** LAV3P | **47 Ag** LAV3P | **48 Cd** LAV3P | **49 In** LAV3P | **50 Sn** LAV3P | **51 Sb** LAV3P | **52 Te** LAV3P | **53 I** LAV3P | **54 Xe** LAV3P |
| **55 Cs** LAV3P | **56 Ba** LAV3P | **57 La** LAV3P | **72 Hf** LAV3P | **73 Ta** LAV3P | **74 W** LAV3P | **75 Re** LAV3P | **76 Os** LAV3P | **77 Ir** LAV3P | **78 Pt** LAV3P | **79 Au** LAV3P | **80 Hg** LAV3P | **81 Tl** LAV3P | **82 Pb** LAV3P | **83 Bi** LAV3P | | | |

# Treatment of Atoms for Pseudospectral LACVP Calculations

***Table D.*** *Calculations that use the LACVP basis set use LACVP effective core potentials for some atoms and LAV3P effective core potentials or 6-31G basis functions (with or without * or ** polarization function options) for other atoms. For instance, a calculation on $CaCl_2$ using the LACVP** basis set would employ 6-31G** basis functions on the chlorine atoms and an LACVP effective core potential for the calcium atom. Any calculation with the LACVP basis set (with or without polarization functions) can be performed pseudospectrally.*

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1 H**<br>6-31G** | | | | | | | | | | | | | | | | | **2 He**<br>6-31G** |
| **3 Li**<br>6-31G** | **4 Be**<br>6-31G** | | | | | | | | | | | **5 B**<br>6-31G** | **6 C**<br>6-31G** | **7 N**<br>6-31G** | **8 O**<br>6-31G** | **9 F**<br>6-31G** | **10 Ne**<br>6-31G** |
| **11 Na**<br>6-31G** | **12 Mg**<br>6-31G** | | | | | | | | | | | **13 Al**<br>6-31G** | **14 Si**<br>6-31G** | **15 P**<br>6-31G** | **16 S**<br>6-31G** | **17 Cl**<br>6-31G** | **18 Ar**<br>6-31G** |
| **19 K**<br>LACVP | **20 Ca**<br>LACVP | **21 Sc**<br>LACVP | **22 Ti**<br>LACVP | **23 V**<br>LACVP | **24 Cr**<br>LACVP | **25 Mn**<br>LACVP | **26 Fe**<br>LACVP | **27 Co**<br>LACVP | **28 Ni**<br>LACVP | **29 Cu**<br>LACVP | **30 Zn**<br>LAV3P | **31 Ga**<br>LAV3P | **32 Ge**<br>LAV3P | **33 As**<br>LAV3P | **34 Se**<br>LAV3P | **35 Br**<br>LAV3P | **36 Kr**<br>LAV3P |
| **37 Rb**<br>LACVP | **38 Sr**<br>LACVP | **39 Y**<br>LACVP | **40 Zr**<br>LACVP | **41 Nb**<br>LACVP | **42 Mo**<br>LACVP | **43 Tc**<br>LACVP | **44 Ru**<br>LACVP | **45 Rh**<br>LACVP | **46 Pd**<br>LACVP | **47 Ag**<br>LACVP | **48 Cd**<br>LAV3P | **49 In**<br>LAV3P | **50 Sn**<br>LAV3P | **51 Sb**<br>LAV3P | **52 Te**<br>LAV3P | **53 I**<br>LAV3P | **54 Xe**<br>LAV3P |
| **55 Cs**<br>LACVP | **56 Ba**<br>LACVP | **57 La**<br>LACVP | **72 Hf**<br>LACVP | **73 Ta**<br>LACVP | **74 W**<br>LACVP | **75 Re**<br>LACVP | **76 Os**<br>LACVP | **77 Ir**<br>LACVP | **78 Pt**<br>LACVP | **79 Au**<br>LACVP | **80 Hg**<br>LAV3P | **81 Tl**<br>LAV3P | **82 Pb**<br>LAV3P | **83 Bi**<br>LAV3P | | | |

# Van der Waals Radii for Solvation Calculations

***Table E.*** *Van der Waals radii (in Angstroms) for some solvation calculations. By default, radii may be reassigned for atoms whose radius values below are shown in bold italics, since Jaguar generally adjusts radii for atoms in certain functional groups. See Section 3.5, section 9.6, and the default.lewis data file for more information on van der Waals radius assignments for solvation calculations. These radius settings can be altered by making* ***vdw2*** *settings in the* ***atomic*** *section of an input file, as described in section 8.2.*

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1 H** | | | | | | | | | | | | | | | | | **2 He** |
| ***1.150*** | | | | | | | | | | | | | | | | | 1.181 |
| **3 Li** | **4 Be** | | | | | | | | | | | **5 B** | **6 C** | **7 N** | **8 O** | **9 F** | **10 Ne** |
| 1.226 | 1.373 | | | | | | | | | | | 2.042 | ***1.900*** | ***1.600*** | ***1.600*** | ***1.682*** | 1.621 |
| **11 Na** | **12 Mg** | | | | | | | | | | | **13 Al** | **14 Si** | **15 P** | **16 S** | **17 Cl** | **18 Ar** |
| 1.491 | 1.510 | | | | | | | | | | | 2.249 | 2.147 | ***2.074*** | 1.900 | ***1.974*** | 1.934 |
| **19 K** | **20 Ca** | **21 Sc** | **22 Ti** | **23 V** | **24 Cr** | **25 Mn** | **26 Fe** | **27 Co** | **28 Ni** | **29 Cu** | **30 Zn** | **31 Ga** | **32 Ge** | **33 As** | **34 Se** | **35 Br** | **36 Kr** |
| 1.906 | 1.700 | 1.647 | 1.587 | 1.572 | 1.511 | 1.480 | 1.456 | 1.436 | 1.417 | 1.748 | 1.381 | 2.192 | 2.140 | 2.115 | 2.103 | ***2.095*** | 2.071 |
| **37 Rb** | **38 Sr** | **39 Y** | **40 Zr** | **41 Nb** | **42 Mo** | **43 Tc** | **44 Ru** | **45 Rh** | **46 Pd** | **47 Ag** | **48 Cd** | **49 In** | **50 Sn** | **51 Sb** | **52 Te** | **53 I** | **54 Xe** |
| 2.057 | 1.821 | 1.673 | 1.562 | 1.583 | 1.526 | 1.499 | 1.481 | 1.464 | 1.450 | 1.574 | 1.424 | 2.232 | 2.196 | 2.210 | 2.235 | ***2.250*** | 2.202 |
| **55 Cs** | **56 Ba** | **57 La** | **72 Hf** | **73 Ta** | **74 W** | **75 Re** | **76 Os** | **77 Ir** | **78 Pt** | **79 Au** | **80 Hg** | **81 Tl** | **82 Pb** | **83 Bi** | | | |
| 2.259 | 1.851 | 1.761 | 1.570 | 1.585 | 1.534 | 1.477 | 1.560 | 1.420 | 1.377 | 1.647 | 1.353 | 2.174 | 2.148 | 2.185 | | | |

# Van der Waals Radii for Charge Fitting

***Table F.*** *Van der Waals radii (in Angstroms) used in calculation of electrostatic potential (ESP) fitted charges. Values for H and for C-Ar are from the DREIDING force field [49]; all other values are from the universal force field [46]. See Electrostatic Potential Fitting in Section 3.6 for more information on computing ESP charges. These radius settings can be altered by making* ***vdw*** *settings in the* ***atomic*** *section of an input file, as described in section 8.9*

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1 H** | | | | | | | | | | | | | | | | | **2 He** |
| 1.597 | | | | | | | | | | | | | | | | | 1.181 |
| **3 Li** | **4 Be** | | | | | | | | | | | **5 B** | **6 C** | **7 N** | **8 O** | **9 F** | **10 Ne** |
| 1.226 | 1.373 | | | | | | | | | | | 2.042 | 1.949 | 1.831 | 1.702 | 1.736 | 1.736 |
| **11 Na** | **12 Mg** | | | | | | | | | | | **13 Al** | **14 Si** | **15 P** | **16 S** | **17 Cl** | **18 Ar** |
| 2.308 | 2.308 | | | | | | | | | | | 2.308 | 2.217 | 2.147 | 2.070 | 1.958 | 1.958 |
| **19 K** | **20 Ca** | **21 Sc** | **22 Ti** | **23 V** | **24 Cr** | **25 Mn** | **26 Fe** | **27 Co** | **28 Ni** | **29 Cu** | **30 Zn** | **31 Ga** | **32 Ge** | **33 As** | **34 Se** | **35 Br** | **36 Kr** |
| 1.906 | 1.700 | 1.647 | 1.587 | 1.572 | 1.511 | 1.480 | 1.456 | 1.436 | 1.417 | 1.748 | 1.381 | 2.192 | 2.140 | 2.115 | 2.103 | 2.095 | 2.071 |
| **37 Rb** | **38 Sr** | **39 Y** | **40 Zr** | **41 Nb** | **42 Mo** | **43 Tc** | **44 Ru** | **45 Rh** | **46 Pd** | **47 Ag** | **48 Cd** | **49 In** | **50 Sn** | **51 Sb** | **52 Te** | **53 I** | **54 Xe** |
| 2.057 | 1.821 | 1.673 | 1.562 | 1.583 | 1.526 | 1.499 | 1.481 | 1.464 | 1.450 | 1.574 | 1.424 | 2.232 | 2.196 | 2.210 | 2.235 | 2.250 | 2.202 |
| **55 Cs** | **56 Ba** | **57 La** | **72 Hf** | **73 Ta** | **74 W** | **75 Re** | **76 Os** | **77 Ir** | **78 Pt** | **79 Au** | **80 Hg** | **81 Tl** | **82 Pb** | **83 Bi** | | | |
| 2.259 | 1.851 | 1.761 | 1.570 | 1.585 | 1.534 | 1.477 | 1.560 | 1.420 | 1.377 | 1.647 | 1.353 | 2.174 | 2.148 | 2.185 | | | |

# Covalent Radii for Bonding Evaluation

***Table G.*** *Covalent radii (in Angstroms) used to evaluate levels of interaction (such as bonds) between atoms. Generally, Jaguar considers two atoms bonded if the distance between them is less than* ***covfac*** *times the sum of their covalent radii, where* ***covfac*** *is keyword with a default value of 1.2. These radius settings can be altered by making* ***cov*** *settings in the* ***atomic*** *section of an input file, as described in sectoin 8.9.*

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1 H** 0.32 | | | | | | | | | | | | | | | | | **2 He** 0.93 |
| **3 Li** 1.23 | **4 Be** 0.90 | | | | | | | | | | | **5 B** 0.82 | **6 C** 0.77 | **7 N** 0.75 | **8 O** 0.73 | **9 F** 0.72 | **10 Ne** 0.71 |
| **11 Na** 1.54 | **12 Mg** 1.36 | | | | | | | | | | | **13 Al** 1.18 | **14 Si** 1.11 | **15 P** 1.06 | **16 S** 1.02 | **17 Cl** 0.99 | **18 Ar** 0.98 |
| **19 K** 2.03 | **20 Ca** 1.74 | **21 Sc** 1.44 | **22 Ti** 1.32 | **23 V** 1.22 | **24 Cr** 1.18 | **25 Mn** 1.17 | **26 Fe** 1.17 | **27 Co** 1.16 | **28 Ni** 1.15 | **29 Cu** 1.17 | **30 Zn** 1.25 | **31 Ga** 1.26 | **32 Ge** 1.22 | **33 As** 1.20 | **34 Se** 1.16 | **35 Br** 1.14 | **36 Kr** 1.12 |
| **37 Rb** 2.16 | **38 Sr** 1.91 | **39 Y** 1.62 | **40 Zr** 1.45 | **41 Nb** 1.34 | **42 Mo** 1.30 | **43 Tc** 1.27 | **44 Ru** 1.25 | **45 Rh** 1.25 | **46 Pd** 1.28 | **47 Ag** 1.34 | **48 Cd** 1.48 | **49 In** 1.44 | **50 Sn** 1.41 | **51 Sb** 1.40 | **52 Te** 1.36 | **53 I** 1.33 | **54 Xe** 1.31 |
| **55 Cs** 2.35 | **56 Ba** 1.98 | **57 La** 1.69 | **72 Hf** 1.44 | **73 Ta** 1.34 | **74 W** 1.30 | **75 Re** 1.28 | **76 Os** 1.26 | **77 Ir** 1.27 | **78 Pt** 1.30 | **79 Au** 1.34 | **80 Hg** 1.49 | **81 Tl** 1.48 | **82 Pb** 1.47 | **83 Bi** 1.46 | | | |

# References

The first eighteen references listed below provide general information about the algorithms used in Jaguar and some of their applications. Their titles are included in the listings, and copies of some of these references are available from Schrödinger upon request. The other listings in this section are referenced throughout this User's Guide.

1. R.A. Friesner, "Solution of Self-Consistent Field Electronic Structure Equations by a Pseudospectral Method," Chem. Phys. Lett. **116**, 39 (1985).

2. R.A. Friesner, "Solution of the Hartree-Fock equations by a pseudospectral method: Application to diatomic molecules," J. Chem. Phys. **85**, 1462 (1986).

3. R.A. Friesner, "Solution of the Hartree-Fock equations for polyatomic molecules by a pseudospectral method," J. Chem. Phys. **86**, 3522 (1987).

4. R.A. Friesner, "An Automatic Grid Generation Scheme for Pseudospectral Self-Consistent Field Calculations on Polyatomic Molecules," J. Phys. Chem. **92**, 3091 (1988).

5. M.N. Ringnalda, Y. Won, and R.A. Friesner, "Pseudospectral Hartree-Fock calculations on glycine," J. Chem. Phys. **92**, 1163 (1990).

6. J.-M. Langlois, R.P. Muller, T.R. Coley, W.A. Goddard III, M.N. Ringnalda, Y. Won, and R.A. Friesner, "Pseudospectral generalized valence-bond calculations: Application to methylene, ethylene, and silylene," J. Chem. Phys. **92**, 7488 (1990).

7. M.N. Ringnalda, M. Belhadj, and R.A. Friesner, "Pseudospectral Hartree-Fock theory: Applications and algorithmic improvements," J. Chem. Phys. **93**, 3397 (1990).

8. Y. Won, J.-G. Lee, M.N. Ringnalda, and R.A. Friesner, "Pseudospectral Hartree-Fock gradient calculations," J. Chem. Phys. **94**, 8152 (1991).

9. R.A. Friesner, "New Methods for Electronic Structure Calculations on Large Molecules," Ann. Rev. Phys. Chem. **42**, 341 (1991).

10. W.T. Pollard and R.A. Friesner, "Efficient Fock matrix diagonalization by a Krylov-space method," J. Chem. Phys. **99**, 6742 (1993).

11. R.P. Muller, J.-M. Langlois, M.N. Ringnalda, R.A. Friesner, and W.A. Goddard III, "A generalized direct inversion in the iterative subspace approach for generalized valence bond wave functions," J. Chem. Phys. **100**, 1226 (1994).

12. R.B. Murphy, R.A. Friesner, M.N. Ringnalda, and W.A. Goddard III, "Pseudospectral Contracted Configuration Interaction From a Generalized Valence Bond Reference," J. Chem. Phys. **101**, 2986 (1994).

13. B.H. Greeley, T.V. Russo, D.T. Mainz, R.A. Friesner, J.-M. Langlois, W.A. Goddard III, R.E. Donnelly, Jr., and M.N. Ringnalda, "New Pseudospectral Algorithms for Electronic Structure Calculations: Length Scale Separation and Analytical Two-Electron Integral Corrections," J. Chem. Phys. **101**, 4028 (1994).

14. J.-M. Langlois, T. Yamasaki, R.P. Muller, and W.A. Goddard, "Rule Based Trial Wavefunctions for Generalized Valence Bond Theory," J. Phys. Chem. **98**, 13498 (1994).

15. D.J. Tannor, B. Marten, R. Murphy, R.A. Friesner, D. Sitkoff, A. Nicholls, M. Ringnalda, W.A. Goddard III, and B. Honig, "Accurate First Principles Calculation of Molecular Charge Distributions and Solvation Energies from Ab Initio Quantum Mechanics and Continuum Dielectric Theory," J. Am. Chem. Soc. **116**, 11875 (1994).

16. R.B. Murphy, M.D. Beachy, R.A. Friesner, and M.N. Ringnalda, "Pseudospectral Localized MP2 Methods: Theory and Calculation of Conformational Energies," J. Chem. Phys. **103**, 1481 (1995).

17. D. Lu, B. Marten, Y. Cao, M.N. Ringnalda, R.A. Friesner, and W.A. Goddard III, "*ab initio* Predictions of Large Hyperpolarizability Push-Pull Polymers: Julolidinyl-n-isoxazolone and Julolidinyl-n-*N,N'*-diethylthiobarbituric acid," Chem. Phys. Lett. **242**, 543 (1995).

18. R.B. Murphy, W.T. Pollard, and R.A. Friesner, "Pseudospectral localized generalized Møller-Plesset methods with a generalized valence bond reference wave function: Theory and calculation of conformational energies," J. Chem. Phys. **106**, 5073 (1997).

19. G. Vacek, J.K. Perry, and J.-M. Langlois, Chem. Phys. Lett. **310**, 189 (1999).

20. F.W. Bobrowicz and W.A. Goddard III, in *Modern Theoretical Chemistry: Methods of Electronic Structure Theory,* H.F. Schaefer III, ed., **3**, Chapter 4 (Plenum, New York, 1977).

21. BIOGRAF manual.

22. MacroModel manual.

23. M.J. Frisch, G.W. Trucks, M. Head-Gordon, P.M.W. Gill, M.W. Wong, J.B. Foresman, B.G. Johnson, H.B. Schlegel, M.A. Robb, E.S. Replogle, R. Gomperts, J.L. Andres, K. Raghavachari, J.S. Binkley, C. Gonzalez, R.L. Martin, D.J. Fox, D.J. DeFrees, J. Baker, J.J.P. Stewart, and J.A. Pople, GAUSSIAN 92 (Gaussian, Inc., Pittsburgh, PA, 1992).

24. Babel version 1.3, copyright (©) 1992-96 W. Patrick Walters and Matthew T. Stahl, All Rights Reserved. (Permission of authors granted to incorporate Babel into Jaguar.)

25. A.D. Becke, J. Chem. Phys. **98**, 1372 (1993).

26. A.D. Becke, J. Chem. Phys. **98**, 5648 (1993).

27. J.C. Slater, *Quantum Theory of Molecules and Solids, Vol. 4: The Self-Consistent Field for Molecules and Solids* (McGraw-Hill, New York, 1974).

28. S.H. Vosko, L. Wilk, and M. Nusair, Can. J. Phys. **58**, 1200 (1980). (The VWN correlation functional is described in the paragraph below equation [4.4] on p. 1207, while the VWN5 functional is described in the caption of Table 5 and on p. 1209.)

29. J.P. Perdew, in *Electronic Structure Theory of Solids,* P. Ziesche and H. Eschrig, eds. (Akademie Verlag, Berlin, 1991); J.P.

Perdew, J.A. Chevary, S.H. Vosko, K.A. Jackson, M.R. Pederson, D.J. Singh, and C. Fiolhais, Phys. Rev. B **46**, 6671 (1992).

30. A.D. Becke, Phys. Rev. A **38**, 3098 (1988).

31. C. Lee, W. Yang, and R.G. Parr, Phys. Rev. B **37**, 785 (1988); implemented as described in B. Miehlich, A. Savin, H. Stoll, and H. Preuss, Chem. Phys. Lett. **157**, 200 (1989).

32. J.P. Perdew and A. Zunger, Phys. Rev. B **23**, 5048 (1981).

33. J.P. Perdew, Phys. Rev. B **33**, 8822 (1986), and Erratum, J.P. Perdew, Phys. Rev. B **34**, 7406 (1986).

34. C. Møller and M.S. Plesset, Phys. Rev. **46**, 618 (1934).

35. S. Sæbø and P. Pulay, Theor. Chim. Acta **69**, 357 (1986).

36. S. Sæbø and P. Pulay, Ann. Rev. Phys. Chem. **44**, 213 (1993).

37. S. Sæbø, W. Tong, and P. Pulay, J. Chem. Phys. **98**, 2170 (1993).

38. J.M. Foster and S.F. Boys, Rev. Mod. Phys. **32**, 300 (1960).

39. J. Pipek and P.G. Mezey, J. Chem. Phys. **90**, 4916 (1989).

40. L.B. Harding and W.A. Goddard III, J. Am. Chem. Soc. **97**, 6293 (1975).

41. E.A. Carter and W.A. Goddard III, J. Chem. Phys. **86**, 862 (1987).

42. T.H. Fischer and J. Almlöf, J. Phys. Chem. **96**, 9768 (1992).

43. H.B. Schlegel, Theor. Chim. Acta **66**, 333 (1984).

44. *CRC Handbook of Chemistry and Physics,* R.C. Weast, ed., 60th edition (CRC Press, Boca Raton, FL, 1979). Dielectric constants for 20 deg. C were used.

45. Water's probe radius is set to 1.40 to reproduce solvation energies properly. All other probe radii are calculated from $r^3 = \frac{3m\Delta}{4\pi\rho}$ ($10^{24}$ $A^3/cm^3$), where $r$ is the solvent probe radius in Angstroms, $m$ is the molecular mass obtained by dividing the molecular weight given in ref. [44] in grams per mole by 6.02 x $10^{23}$, $\Delta$ is the packing density and $\rho$ is the density in $g/cm^3$ at 20 deg. C obtained from ref. [44]. Finding the actual $\Delta$ would require a detailed knowledge of the structure of the liquid. Currently, all $\Delta$ values for these liquids are assumed to be 0.5.

(For FCC lattices, Δ is 0.7405, and for BCC lattices, Δ is 0.6802.)

46. A.K. Rappé, C.J. Casewit, K.S. Colwell, W.A. Goddard, and W.M. Skiff, J. Am. Chem. Soc. **114**, 10024 (1992).

47. L.E. Chirlian and M.M. Francl, J. Comput. Chem. **8**, 894 (1987); R. J. Woods, M. Khalil, W. Pell, S.H. Moffat, and V.H. Smith, Jr., J. Comput. Chem. **11**, 297 (1990).

48. C.M. Breneman and K.B. Wiberg, J. Comput. Chem. **11**, 361 (1990).

49. S.L. Mayo, B.D. Olafson, and W.A. Goddard III, J. Phys. Chem. **94**, 8897 (1990).

50. R.S. Mulliken, J. Chem. Phys. **23**, 1833 (1955).

51. NBO 4.0, E.D. Glendening, J.K. Badenhoop, A.E. Reed, J.E. Carpenter, and F. Weinhold, Theoretical Chemistry Institute, University of Wisconsin, Madison, WI, 1996.

52. J. Baker, A.A. Jarzecki, and P. Pulay, J. Phys. Chem A **102**, 1412 (1998).

53. A.P. Scott and L. Radom, J. Phys.Chem. **100**, 16502 (1996).

54. W.J. Hehre, R.F. Stewart, and J.A. Pople, J. Chem. Phys. **51**, 2657 (1969).

55. W.J. Hehre, R. Ditchfield, R.F. Stewart, and J.A. Pople, J. Chem. Phys. **52**, 2769 (1970).

56. W.J. Pietro, B.A. Levi, W.J. Hehre, and R.F. Stewart, Inorg. Chem. **19**, 2225 (1980).

57. W.J. Pietro, E.S. Blurock, R.F. Hout, Jr., W.J. Hehre, D.J. DeFrees, and R.F. Stewart, Inorg. Chem. **20**, 3650 (1980).

58. J.B. Collins, P. von R. Schleyer, J.S. Binkley, and J.A. Pople, J. Chem. Phys. **64**, 5142 (1976).

59. J.S. Binkley, J.A. Pople, and W.J. Hehre, J. Am. Chem. Soc. **102**, 939 (1980).

60. M.S. Gordon, J.S. Binkley, J.A. Pople, W.J. Pietro, and W.J. Hehre, J. Am. Chem. Soc. **104**, 2797 (1982).

61. W.J. Pietro, M.M. Francl, W.J. Hehre, D.J. DeFrees, J.A. Pople, and J.S. Binkley, J. Am. Chem. Soc. **104**, 5039 (1982).

62. P. Pulay, G. Fogarasi, F. Pang, and J.E. Boggs, J. Am. Chem. Soc. **101**, 2550 (1979).

63. J.D. Dill and J.A. Pople, J. Chem. Phys. **62**, 2921 (1975).

64. R. Ditchfield, W.J. Hehre, and J.A. Pople, J. Chem. Phys. **54**, 724 (1971).

65. W.J. Hehre and J.A. Pople, J. Chem. Phys. **56**, 4233 (1972).

66. J.S. Binkley and J.A. Pople, J. Chem. Phys. **66**, 879 (1977).

67. P.C. Hariharan and J.A. Pople, Theor. Chim. Acta **28**, 213 (1973).

68. W.J. Hehre, R. Ditchfield, and J.A. Pople, J. Chem. Phys. **56**, 2257 (1972).

69. M.M. Francl, W.J. Pietro, W.J. Hehre, J.S. Binkley, M.S. Gordon, D.J. DeFrees, and J.A. Pople, J. Chem. Phys. **77**, 3654 (1982).

70. V.A. Rassolov, J.A. Pople, M.A. Ratner, and T.L. Windus, J. Chem. Phys. **109**, 1223 (1998).

71. T. Clark, J. Chandrasekhar, G.W. Spitznagel, and P. von R. Schleyer, J. Comput. Chem. **4**, 294 (1983).

72. M.J. Frisch, J.A. Pople, and J.S. Binkley, J. Chem. Phys. **80**, 3265 (1984).

73. R. Krishnan, J.S. Binkley, R. Seeger, and J.A. Pople, J. Chem. Phys. **72**, 650 (1980).

74. A.D. McLean and G.S. Chandler, J. Chem. Phys. **72**, 5639 (1980).

75. T.H. Dunning, Jr. and P.J. Hay, in *Modern Theoretical Chemistry: Methods of Electronic Structure Theory,* H.F. Schaefer III, ed., **3**, Chapter 1 (Plenum, New York, 1977).

76. A.K. Rappé and W.A. Goddard, unpublished.

77. T.H. Dunning, Jr., J. Chem. Phys. **90**, 1007 (1989).

78. R.A. Kendall, T.H. Dunning, Jr., and R.J. Harrison, J. Chem. Phys. 96, 6796 (1992).

79. D.E. Woon and T.H. Dunning, Jr., J. Chem. Phys. **98**, 1358 (1993).

80. D.E. Woon, K. A. Peterson, and T.H. Dunning, Jr., unpublished.

81. P.J. Hay and W.R. Wadt, J. Chem. Phys. **82**, 270 (1985).

82. P.J. Hay and W.R. Wadt, J. Chem. Phys. **82**, 284 (1985).

83. P.J. Hay and W.R. Wadt, J. Chem. Phys. **82**, 299 (1985).

84. The LACV3P basis set is a triple-zeta contraction of the LACVP basis set developed and tested at Schrödinger, Inc.

85. T.R. Cundari and W.J. Stevens, J. Chem. Phys. 98, 5555 (1993).

86. T.P. Hamilton and P. Pulay, J. Chem. Phys. **84**, 5728 (1986); P. Pulay, J. Comput. Chem. **3**, 556 (1982); P. Pulay, Chem. Phys. Lett. **73**, 393 (1980).

87. S. Obara and A. Saika, J. Chem. Phys. **84**, 3963 (1986).

88. P.M.W. Gill, M. Head-Gordon, and J.A. Pople, J. Chem. Phys. **94**, 5564 (1990); P.M.W. Gill, M. Head-Gordon, and J.A. Pople, Int. J. Quantum Chem. **S23**, 269 (1989); M. Head-Gordon and J.A. Pople, J. Chem. Phys. **89**, 5777 (1988).

89. R.B. Murphy and R.P. Messmer, J. Chem. Phys. **98**, 10102 (1993).

90. For information on Molden, please see the Molden web site http://www.caos.kun.nl/~schaft/molden/molden.html.

91. MOPAC 6, J.J.P. Stewart, QCPE #455.

92. P. Hohenberg and W. Kohn, Phys. Rev. B **136**, 864 (1964).

93. W. Kohn and L.J. Sham, Phys. Rev. A **140**, 1133 (1965).

94. R.G. Parr and W. Yang, *Density-Functional Theory of Atoms and Molecules* (Oxford, New York, 1989).

95. J.K. Labanowski and J.W. Andzelm, eds., *Density Functional Methods in Chemistry* (Springer-Verlag, Berlin, 1991).

96. R. Colle and O. Salvetti, J. Chem. Phys. **93**, 534 (1990).

97. E. Kraka, Chem. Phys. **161**, 149 (1992).

98. G. Audi and A.H. Wapstra, Nuclear Phys. **A595 4**, 409 (1995).

99. P. Császár and P. Pulay, J. Mol. Struct. **114**, 31 (1984).

100. H.B. Schlegel, J. Comput. Chem. **3**, 214 (1982).

101. M.J.D. Powell, Math. Prog. **1**, 26 (1971).

102. J.M. Bofill, J. Comp. Chem. **15**, 1 (1994).

103. B.A. Murtagh and R.W.H. Sargent, Comp. J. **13**, 185 (1972).

104. R. Fletcher, in *Practical Methods of Optimization* (Wiley, New York, 1987).

105. A. Banerjee, N. Adams, J. Simons, and R. Shepard, J. Phys. Chem. **89**, 52 (1985).

106. P. Culot, G. Dive, V.H. Nguyen, and J.M. Ghuysen, Theor. Chim. Acta **82**, 189 (1992).

107. J. Simons, P. Jorgensen, H. Taylor, and J. Ozment, J. Phys. Chem. **87**, 2745 (1983).

108. M. Häser and R. Ahlrichs, J. Comput. Chem. **10**, 104 (1989); D. Cremer and J. Gauss, J. Comput. Chem. **7**, 274 (1986); J. Almlöf, K. Faegri, Jr., and K. Korsell, J. Comput. Chem. **3**, 385 (1982).

109. C.I. Bayly, P. Cieplak, W.D. Cornell, and P.A. Kollman, J. Phys. Chem. 97, 10269 (1993).

110. A.H. Stroud, *Approximate Calculation of Multiple Integrals* (Prentice-Hall, 1971).

111. V.I. Lebedev, Zh. vychisl. Mat. mat. Fiz. **15**, 48-54 (1975).

112. V.I. Lebedev, Zh. vychisl. Mat. mat. Fiz. **16**, 293-306 (1976).

113. V.I. Lebedev, Sibirsk. Mat. Zh. **18**, 132-142 (1977).

114. V.I. Lebedev, in *Theory of Cubature Formula and Numerical Mathematics* (in Russian), S.L Sobolev, ed. ("Nauka" Sibirsk, Otdel., Novosibirsk, 1980), pages 110-114.

115. C.J. Cramer and D.G. Truhlar, J. Comp. Aided Mol. Design **6**, 629 (1992).

116. B. Marten, K. Kim, C. Cortis, R.A.Friesner, R.B. Murphy, M.N. Ringnalda, D. Sitkoff, and B. Honig, "New Model for Calculation of Solvation Free Energies: Correction of Self-Consistent Reaction Field Continuum Dielectric Theory for Short-Range Hydrogen-Bonding Effects," J. Phys. Chem. **100**, 11775 (1996).

117. D. Chasman, M.D. Beachy, L. Wang, and R.A. Friesner, "Parallel Pseudospectral Electronic Structure. I. Hartree-Fock Calculations," *J. Comp. Chem.* 19, 1017-1029 (1998).

118. M. D. Beachy, D. Chasman, R.B. Murphy, and R.A. Friesner, "Parallel Pseudospectral Electronic Structure. II. Localized Møller-Plesset Calculations, *J. Comp. Chem.* 19, 1030-1038 (1998).

119. Y. H. Jang, L. C. Sowers, T. Cagin, and W. A. Goddard III, J. Phys. Chem. A **105**, 274-280 (2001).

# Index


, . . . . . 284
.atomig file . . . . . 86, 159, 196, 237, 242–243
specified in input file . . . . . 159–160
.basis file . . . . . 86, 159, 237, 238–242
specified in input file . . . . . 159–160
.cutoff file . . . . . 86, 89, 123, 159, 237, 253–255
description and format . . . . . 253–255
specified in input file . . . . . 159–160
.daf file . . . . . 86, 159, 237, 244–249
description and format . . . . . 244–249
neighbor ranges . . . . . 110, 244, 245–246
references to content in input file . . . . . 207
specified in input file . . . . . 159–160
.grid file . . . . . 86, 89, 123, 159, 237, 249–252
description and format . . . . . 249–252
references to contents in input file . . . . . 207
specified in input file . . . . . 159–160
.in file—see Input file
.lewis file . . . . . 159, 237, 255–265
description and format . . . . . 255–265
specified in input file . . . . . 159–160
.trn file—see GVB2P5
.trn file


## A


About button . . . . . 34, 125
Accuracy level . . . . . 63–64, 67
input keyword for . . . . . 193
AIMPAC .wfn file, input keyword for . . . . . 204
All-analytic calculation . . . . . 64, 194
output from . . . . . 108
Analytic corrections . . . . . 64, 141–142
.cutoff file determination of . . . . . 255
input keyword for . . . . . 193
Analytic frequencies . . . . . 53
Analytic gradient of energy . . . . . 66, 68, 96, 226
convergence criteria based on . . . . . 68
input keywords for . . . . . 182
der1a and der1b programs . . . . . 226
input keyword for . . . . . 177
lmp2der, lmp2gda, and lmp2gdb programs . . . . . 226
output of maximum and RMS elements for geometry optimization . . . . . 96
Anions . . . . . 16, 168
Atom labels . 9, 11, 22, 43, 45, 86, 105, 109, 120, 162, 206, 207
Atom numbers . . . . . 43, 44, 45
Atomic charges—see ESP fitting, Mulliken population analysis Molecular charge
atomic input file section . . . . . 214
Atomic mass keyword . . . . . 167
Atomic masses
for frequency calculations . . . . . 53
input keyword for . . . . . 167–168
setting in atomic section . . . . . 214–216
Atomic orbital space
printing Fock matrix in . . . . . 117
input keyword for . . . . . 205
printing orbitals in (each SCF iteration) . . . . . 118
input keyword for . . . . . 206
Atomic properties, setting in atomic section 161, 214–219
atomic section
format . . . . . 214
input types . . . . . 216
Atomic units . . . . . 162, 163, 167, 222
Atomic units in output
for geometry . . . . . 109
for multipole moments . . . . . 109


## B


Babel
using to generate output . . . . . 199–201
using to read input files . . . . . 17–19
basgss basis set label . . . . . 220, 221
Basis functions . . . . . 105, 111–114, 243
contracted . . . . . 111, 112, 113
counterpoise calculations . . . . . 15
derivatives of
Gaussian function list in output . . . . . 114, 201
evaluating contributions on grid points . . . . . 244
file containing . . . . . 237, 238–242
for individual atoms . . . . . 217, 218
Gaussian function list in output . . . . . 105, 111–114
input keyword for . . . . . 201
number of . . . . . 86
type, as listed in output . . . . . 105, 120, 206, 207
uncontracted . . . . . 111, 112
basis input type . . . . . 217, 218
Basis set . . . . . 56–59, 86, 111–114, 242
diffuse functions . . . . . 56, 57, 190, 239
for individual atoms . . . . . 217, 218
Gaussian function list in output . . . . . 105, 111–114
input keyword for . . . . . 201
included in generated GAUSSIAN 92 input file . . 136
input keyword for . . . . . 190–191, 221
keywords . . . . . 190
minimal, with GVB . . . . . 45
output from . . . . . 108
polarization functions . . . . 56, 56–57, 59, 190, 239

specifying for GAUSSIAN 92 input .....138, 204
specifying for initial guess ..... 61–62, 220, 221
Basis set superposition error ..... 15, 41, 42, 149
Basis Set window .....56–59
output from ..... 108
Basis sets
file containing ..... 237, 238–242
Batch input file
example ..... 233
format ..... 230
Batch jobs
jaguar batch command for ..... 158, 230–235
running from Jaguar interface .....29–31
Benzene ..... 48
BFGS method for Hessian updating
input keyword for ..... 180
BIOGRAF
.bgf files ..... 21
reading from Jaguar interface ..... 273
.hes files ..... 219
reading from Jaguar interface ..... 273
reading files from Jaguar interface .....17–19
Bohr units for geometry input ..... 162, 163, 167, 222
Bond dissociation ..... 46, 130, 230
Bond lengths and angles output option ..... 94, 109–110
input keyword for ..... 202
Bond lengths, bond angles, or torsions
freezing ..... 69
Bonding types, describing in lewis files ..... 257
Boys localization .....64, 226
Fock matrix output in Boys orbital space
input keyword for ..... 202
input keywords for ..... 172, 173, 197, 198
orbital printing ..... 119
input keyword for .....198, 206
order ..... 108
output from ..... 108


## C


Calculating molecular properties ..... 48
Calculation host ... 4, 27, 125, 126, 126, 155, 157, 266, 271–272
Calculation options, output file changes for ..... 90
Canonical orbital space
printing Fock matrix in ..... 117
input keyword for ..... 205
printing orbitals in (each SCF iteration) ....118, 120
input keyword for ..... 206
Carbon tetrachloride ..... 48
Cartesian format ..... 8, 9–11, 160, 162–163
Cations .....16, 168
ch program ..... 226
output from ..... 98, 102–106
for solvation .....98, 101
Charge fitting—see ESP fitting, Mulliken population analysis
charge input type ..... 218
Charge keywords ..... 167
chdens electron density output file ..... 51, 104, 187
Check Job button ..... 122
Check option ..... 7, 32, 33, 122
Chlorobenzene ..... 48
CI—see Configuration interaction
Cleanup button ..... 23
Comment line ..... 28, 32, 160
Configuration interaction (CI)
coefficients
for GVB pair natural orbitals, in output ..... 92
energy lowering
for GVB pair natural orbitals, in output ..... 93
connect input file section ..... 165
Connectivity ..... 21
connectivity table output option ..... 110
input keyword for ..... 202
input keyword for bonding .....110, 167
Consecutive Jaguar jobs
running from Jaguar interface .....29–31
running with jaguar batch .....230–235
Constraints for geometry optimization 10–11, 14, 69–70, 70–71, 164
input keyword for applying ..... 179
input keywords for ..... 179
output from ..... 94
Contracted basis functions ..... 111, 111, 112, 113
Convergence criteria
geometry optimization ..... 68, 96, 97
analytic gradient criteria .....68, 96
displacement criteria .....68, 96
energy change criterion ..... 96
gradient-related only ..... 179
input keywords for ..... 178, 181–183
SCF energy criterion ..... 67
SCF energy criterion .....63, 132
input keyword for ..... 191
solvation energy ..... 101
input keyword for ..... 185
Convergence problems, troubleshooting ..... 274
Convergence schemes ..... 62
DIIS .....62, 108
input keywords for .....191–192
OCBSE ..... 62
output from ..... 108
coord input file section ..... 165
Coordinates
constraining ..... 69
freezing ..... 70
Coulomb and exchange matrices output option ..... 117
input keyword for ..... 205
Coulomb corrections ..... 255
Coulomb energy contribution ..... 89, 92, 108, 116
Coulomb field, charge fitting to ..... 103
Coulomb operator (J) .....116, 193
input keyword for output ..... 205
obtaining i/o information for ..... 109
Per Iteration matrix output option ..... 117
pseudospectral assembly of .....141–142
Counterpoise calculations ..... 15, 163, 217, 218

Coupled perturbed Hartree-Fock (CPHF) terms
for LMP2 dipole moments . . . . . 42, 50
for LMP2 ESP fitted charges . . . . . 42, 49
for polarizability and hyperpolarizability calculations 50
Coupled perturbed Hartree-Fock—see CPHF calculations
cov input type . . . . . 216
CPHF calculations
grid used for . . . . . 208
of hyperpolarizability . . . . . 50, 186
of polarizability . . . . . 50, 186
cpolar program . . . . . 226
cpu time . . . . . 109, 201
Culot-Fletcher method for trust radius adjustment
input keyword for . . . . . 181
cut10 . . . . . 86
cut20 . . . . . 86
Cutoff methods . . . . . 63–64, 67, 193
Cutoffs . . . . . 193, 195, 253–255
.cutoff file . . . . . 86, 237, 253–255
as shown in output . . . . . 89, 123
Cyclohexane . . . . . 48

# D

daf input type . . . . . 218
Data directory . . . . . 237
Dealiasing function keywords . . . . . 207
Dealiasing functions . . . . . 57, 86, 201, 237, 244–249
contracted . . . . . 244, 246, 247
dealiasing function choices for calculation . 110, 161, 207–209, 245, 246
Gaussian function list in output
input keyword for . . . . . 201
long-range . . . . . 244, 246, 247
neighbor ranges for . . . . . 110, 244, 245–246
short-range . . . . . 244, 245, 247
uncontracted . . . . . 244, 246, 247
default.atomig file . . . . . 242
default.basis file . . . . . 238
Delocalization of LMP2 pairs . . . . . 172, 213–214
Density
plotting with plot section . . . . . 228–229
Density functional theory (DFT) . . . . . 38–41, 152–153
customized functional combinations for . . . . . 40–41
hybrid methods . . . . . 39, 40
input keywords for . . . . . 173–177, 207
optimization output . . . . . 93–97
output from . . . . . 90
standard functional combinations for . . . . . 39–40
Density matrix . . . . . 88, 122
convergence criterion for elements . . . . . 63, 67
input keyword for . . . . . 191
in DIIS error vector . . . . . 89, 123
input keyword for output . . . . . 205
Per Iteration output option . . . . . 116–117
Density matrix output option . . . . . 116–117
input keyword for . . . . . 205
der1a program . . . . . 226
output from . . . . . 94, 95, 97
der1b program . . . . . 226
output from . . . . . 94, 95, 97
Derivatives of basis set functions . . . . . 114
Gaussian function list in output . . . . . 114
input keyword for . . . . . 201
Derivatives of energy . . . . . 66
Detailed timing information output option . . . . . 109
input keyword for . . . . . 201
DFT keywords . . . . . 173
DFT window . . . . . 38–41
output from . . . . . 90
DFT—see Density functional theory
Dichloroethane . . . . . 48
Dielectric constant . . . . . 47
input keywords for . . . . . 184
Dielectric continuum method—see pbf program
Differential density matrix . . . . . 88, 122
input keyword for output . . . . . 205
Per Iteration output option . . . . . 116–117
RMS of elements in output . . . . . 89, 123
Differential density matrix output option . . . . . 116–117
input keyword for . . . . . 205
DIIS coefficients output option
input keyword for . . . . . 205
DIIS convergence scheme . . . . . 62, 88, 108, 123
coefficients, listing in output
input keyword for . . . . . 205
input keywords for . . . . . 191–192
DIIS error vector
maximum elements, listed in output . . . . . 89, 123
Dimming of menu items . . . . . 32
Dipole moment—see Multipole moments
Direct Inversion in the Iterative Subspace method—see DIIS convergence scheme
Displacement . . . . . 68
convergence criteria based on . . . . . 68
input keywords for . . . . . 183
Display host . . . . . 20, 266, 269–270
Display window . . . . . 20
Displaying geometry input . . . . . 19–23
Dissociation . . . . . 46, 130, 230
Driver for geometry scan . . . . . 78–79
dsolv program . . . . . 227
Dummy atoms . . . . . 14, 219

# E

Echo input file and parameter list output option . . 108–109
echo input file section . . . . . 225
ECPs—see Effective core potentials
Edit window . . . . . 8–9, 44
counterpoise calculations, setting up . . . . . 15
fixing bond lengths or angles . . . . . 14
fixing Cartesian coordinates . . . . . 10–11
freezing bond lengths or angles . . . . . 69–70, 70–71
freezing Cartesian coordinates . . . . . 70–71
Editing a geometry . . . . . 8–9
Editing input files from interface . . . . . 34–35

Effective core potentials (ECPs) . . 57–59, 225, 240–242, 243
efields input file section . . . . . 222
Eigenvector following in transition state optimizations . 76
input keywords for . . . . . 181
Eigenvectors and eigenvalues of overlap matrix output option
input keyword for . . . . . 202
elden program . . . . . 226
Electric field for polarizability calculations . . . . . 188
input file section for . . . . . 161, 222–223
Electron density . . . . . .51, 226
input keywords for . . . . . 187, 188, 208
output from calculation . . . . . 104
Electron-nuclear energy from solvation calculations . . 100
Electrons, information in output . . . . . 88, 95, 97
electrostatic potential . . . . . 188
Electrostatic potential fitting—see ESP fitting
Electrostatic properties
ch program . . . . . 226
Energy components output option . . . . . 116
input keyword for . . . . . 205
Energy convergence criterion . . . . . .63, 132
input keyword for . . . . . 191
Energy difference
geometry optimization convergence criterion based on 68, 96
input keyword for . . . . . 183
Energy output . . . . . .88–90
nuclear repulsion . . . . . .86, 89, 99, 100, 116
nuclear-point charge energy from solvation calculations . . . . . 99
SCF . . . . . .88–90
components for each iteration . . . . . 116
components, for each iteration . . . . . 205
Coulomb contribution . . . . . 89, 92, 108, 116
exchange contribution . . . . . 89, 92, 108, 116
for each iteration . . . . . .89, 123
one-electron . . . . . 89, 98, 100, 116
orbital energies . . . . . 116
total . . . . . .89, 116
two-electron . . . . . .89, 116
solvation . . . . . 98–101, 116
electron-nuclear energy . . . . . 100
first shell correction factor . . . . . 101
nuclear-solvent energy . . . . . 100
reorganization energy . . . . . 101
solute cavity energy . . . . . .99, 101
solution phase energy . . . . . .100–101
solvation energy . . . . . 101
total solvent energy . . . . . 100
Enthalpy calculations—see Thermochemical properties
Entropy calculations—see Thermochemical properties
Environment variables, setting . . . . . 267
ESP . . . . . 204
ESP fitting . . . . . .49–50
ch program . . . . . 226
constraining to reproduce multipole moments 49, 103, 131
output from . . . . . 104
error as reported in output . . . . . 103
for LMP2 wavefunctions . . . . . 49
for solvation calculations . . . . . .47, 98
grid for . . . . . .49–50
input keywords for . . . . . .188, 209
input keywords for . . . . . 185
output from . . . . . .103–104
recalculating multipole moments from . . . . . .49, 50
output from . . . . . 104
to atom centers . . . . . 49
output from . . . . . 103
to atom centers and bond midpoints . . . . . 49
output from . . . . . 103
esp input type . . . . . 216
Exchange corrections . . . . . 255
Exchange energy contribution . . . . . 89, 92, 108, 116
Exchange operator (K) . . . . . .116, 193
input keyword for output . . . . . 205
obtaining i/o information for . . . . . 109
Per Iteration matrix output option . . . . . 117
pseudospectral assembly of . . . . . .141–142
Executable directory . . . . . .86, 156, 157, 158, 159
Execution path . . . . . 161, 225–228

# F

Field, electric, input file section for . . . . . 161, 222–223
File names . . . . . .27, 31
File viewer window . . . . . .122–123
Files output keywords . . . . . 203
Files output options . . . . . .114–115
GAMESS input file (.gamess) generation . . . . . 115
GAMESS input file (gamess.inp) generation . . . . 204
GAUSSIAN 92 basis set (.gbs file) output option 115, 138
GAUSSIAN 92 input deck (.g92 file) generation 114–115, 136
input keywords corresponding to . . . . . .203–204
Molden orbitals file (.molf file) output option . . . 115
SPARTAN archive file (.arc file) generation . . . . 115
XYZ file (.xyz file) output option . . . . . 115–??
First shell correction factor for solvation . . . . . 101
input keyword for . . . . . 184
Fixing bond lengths or angles for geometry optimization . 14, 164
input keyword for applying . . . . . 179
input keywords for . . . . . 179
output from . . . . . 94
Fixing Cartesian coordinates for geometry optimization . . 10–11
Fock matrix . . . . . .88, 123
in DIIS error vector . . . . . .89, 123
input keywords for output . . . . . 205
new estimate from DIIS scheme . . . . . 88, 117, 123
output in Boys orbital space
input keyword for . . . . . 202
Per Iteration output options . . . . . 117

pseudospectral assembly of . . . . . 139–142
updating . . . . . 88, 117, 122
input keyword for . . . . . 194
Fock matrix output option
in AO (HF) or MO (GVB) space . . . . . 117
input keyword for . . . . . 205
in CO space . . . . . 117
input keyword for . . . . . 205
Forces . . . . . 66, 177, 226
analytic calculation of . . . . . 66
numerical calculation of . . . . . 66
input keywords for . . . . . 178, 182
output from . . . . . 97–98
numerical calculation of derivatives of . . . . . 226
input keywords for . . . . . 182, 189
output from . . . . . 106–107
table in output . . . . . 94
formal input type . . . . . 216
frag input type . . . . . 218
Freezing bond lengths or angles for geometry optimization 69–70, 70–71
Freezing Cartesian coordinates for geometry optimization 70–71
freq program . . . . . 226
Frequencies . . . . . 53
for a fragment of system . . . . . 218–219
freq program . . . . . 226
input keywords for . . . . . 182, 189
output . . . . . 106–107
scaling . . . . . 54
input keywords for . . . . . 189
Frequencies window . . . . . 52–56
output from . . . . . 106–107
Frequency-related keywords . . . . . 188
Frequency-related properties . . . . . 54–56
output . . . . . 106–107
Functionals settings . . . . . 40


# G


GAMESS
generating input files with Jaguar . . . . . 115
input files, generating with Jaguar
input keyword for . . . . . 204
Gas phase optimizations . . . . . 48
GAUSSIAN
input files
reading from Jaguar interface . . . . . 17–19
GAUSSIAN 92
basis set output (.gbs file) generating with Jaguar . 204
basis set output (.gbs file), generating with Jaguar . . 115, 138
Hessian format . . . . . 219
input files
generating with Jaguar 114–115, 136–138, 203, 204
reading from Jaguar interface . . . . . 138, 273
orbital output in format for . . . . . 120, 206
GAUSSIAN 92 basis set (.gbs file) output option .115, 138
input keyword for . . . . . 204
GAUSSIAN 92 input deck (.g92 file) generation 114–115, 136
input keyword for . . . . . 137, 203, 204
Gaussian function list in output
for basis set . . . . . 105, 111–114
input keyword for . . . . . 201
for dealiasing functions
input keyword for . . . . . 201
for derivatives of basis functions . . . . . 114
input keyword for . . . . . 201
gen input file section . . . . . 166
Generalized gradient approximation . . . . . 153
Generalized Valence Bond method—see GVB calculations
Geometries in atomic units also output option . . . . . 109
Geometry buttons . . . . . 19–26, 44
Display option . . . . . 19–23
Symmetrize option . . . . . 25–26, 160
Geometry input . . . . . 5–6, 8–16, 16–19
Cartesian format . . . . . 8, 9–11, 162–163
displaying . . . . . 19–23
editing . . . . . 8–9
file types for scanning . . . . . 17–19
format . . . . . 9–16, 162–166
input file sections for . . . . . 162–166
keywords . . . . . 167
obtaining from MOPAC . . . . . 134
output file, echoed in . . . . . 86
symmetrizing . . . . . 25–26, 160, 273
tolerance . . . . . 25
translating and rotating during calculation . . . . . 86
troubleshooting . . . . . 273
units
input keyword . . . . . 162, 163, 167, 222
Z-matrix format . . . . . 8, 11–14, 163–165
Geometry optimization . . . . . 66–79
calculating forces only . . . . . 66
der1a and der1b programs . . . . . 226
input keyword for . . . . . 177
constraining bond lengths or angles .14, 69–70, 70–71, 164
input keywords for . . . . . 179
output from . . . . . 94
constraining Cartesian coordinates . . . 10–11, 70–71
convergence criteria . . . . . 68, 96, 97
gradient-related only . . . . . 179
input keywords for . . . . . 178, 181–183
convergence criterion for SCF . . . . . 67
detailed output . . . . . 110
fixed bond lengths or angles for . . . . . 14, 164
input keywords for . . . . . 179
output from . . . . . 94
fixed Cartesian coordinates for . . . . . 10–11
frozen bond lengths or angles for . . . . 69–70, 70–71
frozen Cartesian coordinates for . . . . . 70–71
GDIIS method
input keyword for . . . . . 179
generating input with new geometry . . . . . 130
geopt program . . . . . 226

in solution . . . . . 46
output from . . . . . 102
without gas phase calculation . . . . . .48, 177
initial Hessians for . . .15–16, 68, 77–78, 106, 134–135, 163–164
input file section . . . . . 161, 219–220
input keyword for . . . . . 179
input keywords for . . . . . .177–183
limiting step size for . . . . . .69, 185
input keywords for . . . . . .181, 182
maximum number of iterations for convergence . . 67, 97, 131
input keyword for . . . . . 178
output from . . . . . 93–98, 102
bond lengths and angles . . . . . 94, 109–110
forces table . . . . . 94
refinement of initial Hessian for 15–16, 76–77, 77–78, 163–164, 180
tips . . . . . .130–131
troubleshooting . . . . . 274
trust radius for . . . . . .69, 185
input keywords for . . . . . .181, 182
updating of Hessian during
input keyword for . . . . . 180
Geometry optimization keywords . . . . . 177
Geometry scans . . . . . .78–79
Geometry-only file
reading from interface . . . . . 273
Geometry—see Geometry input, Geometry optimization
geopt program . . . . . 226
output from
for geometry optimization . . . . . .95–97
Ghost atoms
use in charge fitting to bond midpoints . . . . . 103
Gibbs free energy calculations—see Thermochemical properties
GPTSFILE . . . . . 188
Gradient—see Analytic gradient of energy
Grasp program . . . . . 187
grid input type . . . . . 218
Grid keywords . . . . . 207
grid program . . . . . 225
output from . . . . . 87, 95, 97
Grid shell locations output option
input keyword for . . . . . 201
Grids . . . . . 57, 63–64, 67, 193, 194, 237, 249–252
dealiasing functions for fitting to . . . . . .244–249
evaluating basis function contributions on . . . . . 244
for DFT calculations . . . . . 38
for ESP fitting . . . . . .49–50
input keywords for . . . . . .188, 209
grid choices for calculation .161, 207–209, 249, 254
specifying your own grid . . . . . 159, 160, 209
grid generation with program grid . . . . . 225
output about grids . . . . . 86
number of grid points . . . . . 87
number of grid points for charge fitting . . . . 103
which used for iterations . . . . . .89, 123
shells for . . . . . .250–252
ultrafine only . . . . . 64
guess input file section . . . . . .61, 220
GVB calculations . . . . . .44–45, 117, 142–148
from HF converged wavefunction . . . . . .62, 197
from HF initial guess . . . . . 62
from input HF wavefunction . . . . . 197
generating GAUSSIAN 92 input for . . . . . 137
gvbig program for GVB initial guess . . . . . 226
input keywords for SCF settings . . . . . .191–195
optimization output . . . . . .93–97
output from . . . . . 91–93, 116
pair selection tips . . . . . 130
Per Iteration output option for data . . . . . 117
printing orbitals . . . . . 119, 120, 206
after initial guess is generated . . . . . .118, 206
after SCF iterations . . . . . .119, 206
at end of job . . . . . 119
scf program . . . . . 226
troubleshooting . . . . . 274
GVB data output option . . . . . 117
input keyword for . . . . . 205
gvb input file section . . . . . 212
GVB keywords . . . . . 169
GVB pairs . . . . . .44–45
displaying . . . . . 21–22, 44
for GVB-LMP2 calculations . . . . . 46
input file section for . . . . . 161, 212–213
output information . . . . . 92, 92–93
CI energy lowering . . . . . 93
natural orbital assignment . . . . . 92
natural orbital CI coefficients . . . . . 92
orbital overlap . . . . . 93
selection tips . . . . . 130
setting from Lewis dot structure . . . . . .169–171
specifying for GAUSSIAN 92 input generation . . 137
troubleshooting . . . . . 274
GVB window . . . . . .44–45
output from . . . . . .91–93
GVB2P5
.trn file
generating with Jaguar . . . . . 204
convergence scheme (OCBSE) . . . . . 62
generating MQM basis set (.bas file) for
input keyword for . . . . . 204
orbital output in format for . . . . . .120, 206
GVB2P5 trans matrix (.trn file) output option
input keyword for . . . . . 204
gvbig program . . . . . .92, 226
GVB-LMP2 calculations . . . . . .45–46
GVB-RCI calculations . . . . . 44–45, 146–148
optimization output . . . . . .97–98


# H


ham input file section . . . . . 223
Hamiltonians
information in output . . . . . 88, 92, 116
user input of . . . . . .161, 223
Harmonic frequencies . . . . . 107

Hartree-Fock (HF) calculations .....44, 85, 86, 89, 123
input keywords for SCF settings .....191–195
printing orbitals
after initial guess is generated ... 118, 120, 206
after SCF iterations .....119, 206
at end of job ..... 119
scf program ..... 226
used to get GVB initial guess ..... 62
output from ..... 93
Hartree-Fock calculations
optimization output .....93–97
output from standard ..... 85
Heat capacity calculations—see Thermochemical properties
Help window .....34, 273
hess input file section ..... 219
Hessian refinement, specifiying coordinates for ..... 77
Hessians ..... 68, 76–77
effect of quality on geometry convergence ....67, 76
from MOPAC .....134–135
input file section for input of ..... 161, 219–220
level shifting ..... 181
input keyword for ..... 180
reading BIOGRAF Hessians from Jaguar interface 19
refinement of initial Hessian .15–16, 76–77, 77–78, 163–164
input keywords for ..... 180
selecting initial Hessian .....68, 106
input keyword for ..... 179
updating
input keyword for ..... 180
Heteroatom pairs for local LMP2 calculations ....43, 171
energy output for .....91, 202
hfig program ..... 225
output from .....86–87
How: Orbitals output options ..... 119–122, 221
input keywords for .....206–207
Hybrid methods ..... 40
Hybridization types
describing in Lewis files ..... 258
Hyperpolarizability ..... 50–51, 187, 226
input keywords for ..... 186, 187, 188
output from ..... 104

# I

Icon for Jaguar interface ..... 127
imw ..... 189
Infrared intensities .....54–55
input keyword for ..... 189
output ..... 107
Initial guess .....61–62, 86, 128–130
file information for ..... 237, 242–243
for transition-metal-containing systems 61, 128–130
GVB initial guess
from HF converged wavefunction ..... 197
from input HF wavefunction ..... 197
gvbig program ..... 226
input keyword for output ..... 205
Per Iteration output option ..... 117
HF initial guess program hfig ..... 225
input file section for ..... 161, 220–222
input keyword for .....196, 220
orbital output in format for .....120, 206
stopping after .....136, 196
Initial guess keywords ..... 195
Input file 4, 17–19, 26, 28, 31–32, 37, 66, 80, 159–229, 237
atomic section ..... 98, 129–130, 161, 214–219
defining fragments with .....218–219
connect section ..... 161, 165–166
coord section ..... 161, 165–166
description and format .....159–162
description of sections .....160–162
echo section .....161, 225
echoing in output .....161, 225
echoing in output file .....108–109
editing from interface .....34–35
efields section ..... 161, 222–223
gen section . 89, 123, 137, 138, 161, 166–212, 245, 246, 249, 253
generating with Save window ..... 31–32, 158
guess section ..... 61, 120, 161, 206, 220–222
gvb section ..... 161, 212–213
ham section .....161, 223
hess section ..... 161, 219–220
keywords .....166–212
lmp2 section .....161, 171, 172, 213–214
nbo section ..... 52, 161, 229–230
new ..... 131–133, 212
orbman section ..... 161, 223–224
path section ..... 161, 225–228
plot section ..... 161, 228–229
pointch section .....161, 222
reading from interface ..... 17–19, 273
section delineators ..... 162
spacing characters ..... 160
submitting by hand .....156–158
zmat section ..... 160, 162–164
zmat2 section .....162–164
zmat3 section .....162–164
zvar section ..... 160, 163, 164–165
zvar2 section .....164–165
zvar3 section .....164–165
Input file directory ..... 31
Input of molecular structure (geometry)—see Geometry input
Input type
basis ..... 218
charge ..... 218
cov ..... 216
daf ..... 218
esp ..... 216
grid ..... 218
isotope ..... 216
mass ..... 216
mulk ..... 216
multip ..... 216
vdw ..... 216

vdw2 . . . . . 216
input type
formal . . . . . 216
Integrals
one-electron . . . . . 86
calculation with program onee . . . . . 225
energy contribution listed in output . . . . . 89
two-electron . . . . . 64
energy contribution listed in output . . . . . 89, 108
energy output breakdown into J and K parts . 92, 116
Interface
customizing appearance with jaguar.style file . . . . 127
customizing settings with jaguar.hosts file . 124–126, 274–275
problems starting . . . . . 268–270
Interface host . . . . . 4, 269–270, 271–272
Internal coordinates . . . . . 178
specifying in input with connect section . . . . 165–166
specifying in input with coord section . . . . . 165–166
Ions . . . . . 16, 168
ip172 . . . . . 188, 204
IR intensities . . . . . 54–55
input keyword for . . . . . 189
output . . . . . 107
ira program . . . . . 226
irb program . . . . . 226
isotope input type . . . . . 216
Isotopes
choosing with atomic section . . . . . 214–216
input keyword for . . . . . 167–168
setting masses in atomic section . . . . . 214–216
Iterations, geometry convergence, maximum number for . 67, 97, 131
input keyword for . . . . . 178
Iterations, SCF, maximum number of . . . . . 62, 128
input keyword for . . . . . 192


# J


jaguar command . . . . . 5, 154–159, 268
jaguar batch . . . . . 158, 230–235
jaguar help . . . . . 155
jaguar hosts . . . . . 155
jaguar jobs . . . . . 155
jaguar kill . . . . . 158–159
jaguar run . . . . . 156–158
host option . . . . . 157–158
save temporary files option . . . . . 157
time stamps to log file option . . . . . 157
version option . . . . . 158
jaguar run,
nice option . . . . . 157
jaguar versions . . . . . 156
Jaguar copyright information . . . . . 86
Jaguar files
.atomig file—see .atomig file
.basis file—see .basis file
.cutoff file—see .cutoff file
.daf file—see .daf file
.grid file—see .grid file
.lewis file—see .lewis file
input file—see Input file
log file—see log file
output file—see Output file
Jaguar programs
ch—see ch program
cpolar—see cpolar program
der1a—see der1a program
der1b—see der1b program
dsolv—see dsolv program
elden—see elden program
freq—see freq program
geopt—see geopt program
grid—see grid program
gvbig—see gvbig program
hfig—see hfig program
ira—see ira program
irb—see irb program
jexec—see jexec program
lmp2der—see lmp2der program
lmp2dip—see lmp2dip program
lmp2gda—see lmp2gda program
lmp2gdb—see lmp2gdb program
lmp2—see lmp2 program
loclmp2—see loclmp2 program
locpost—see locpost program
machid—see machid program
nude—see nude program
onee—see onee program
pbf—see pbf program
polar—see polar program
post—see post program
pre—see pre program
probe—see probe program
rci—see rci program
rwr—see rwr program
scf—see scf program
sole—see sole program
solv—see solv program
timex—see timex program
jaguar results command . . . . . 80–85
jaguar.hosts file . 27, 124–126, 155, 237, 268–269, 272, 274–275
jaguar.style file . . . . . 127
jexec program . . . . . 225
Job directory, local . . . . . 27, 32
Job name . . . . . 27, 31, 33, 86, 159
Job Status window . . . . . 7, 28, 33


# K


Keyword settings
input keyword for echoing in output . . . . . 202
Keywords
atomic mass . . . . . 167, 168
dealiasing function . . . . . 207
DFT . . . . . 173

files output . . . . . 203
frequency-related . . . . . 188
geometry optimization . . . . . 177
grid . . . . . 207
GVB . . . . . 169
initial guess . . . . . 195
lewis dot structure . . . . . 169
LMP2 . . . . . 171
localization . . . . . 197
memory usage . . . . . 209
orbital output . . . . . 205
properties . . . . . 185
SCF iteration . . . . . 204
SCF method . . . . . 191
solvation . . . . . 183
standard output . . . . . 201
transition state . . . . . 177
Killing jobs . . . . . 158–159


# L


LDA (local density approximation) . . . . . 39
Least-squares operator Q, description of . . . . . 140
Level shifting . . . . . 63, 181
input keyword for . . . . . 180
to help SCF convergence . . . . . 63
Lewis dot structure
.lewis file . . . . . 237, 255–265
input keywords for . . . . . 169–171
setting GVB pairs from . . . . . 169–171
setting van der Waals radii from . . 98, 184, 255–265
Lewis dot structure keywords . . . . . 169
Linear synchronous transit (LST) methods—see QST-guided transition state searches
LMP2 calculations
setting up . . . . . 43
lmp2 input file section . . . . . 213
LMP2 keywords . . . . . 171
LMP2 method . . . . . 41–43, 148–152, 226
applying counterpoise corrections with . . . . . 15
description of pseudospectral implementation . . 148–152
grid used for . . . . . 208
input keywords for . . . . . 171
output from . . . . . 91
LMP2 pairs
delocalization of . . . . . 172, 213–214
input file section for . . . . . 161, 171, 172, 213–214
input keywords for . . . . . 171, 172
lmp2 program . . . . . 226
output from . . . . . 91
LMP2 window
output from . . . . . 91
lmp2der program . . . . . 226
lmp2dip program . . . . . 226
lmp2gda program . . . . . 226
lmp2gdb program . . . . . 226
output from . . . . . 94
Local density approximation (LDA) . . . . . 39, 153
Local job directory . . . . . 27, 32
Local local MP2 method . . . . . 42–43, 43
grid used for . . . . . 208
input file section for . . . . . 161, 213–214
input keywords for . . . . . 91, 171, 202
Local MP2 method . . . . . 41–43, 148–152, 226
applying counterpoise corrections with . . . . . 15
description of pseudospectral implementation . . 148–152
grid used for . . . . . 208
input keywords for . . . . . 171
output from . . . . . 91
Local MP2 settings . . . . . 41
Local MP2 window . . . . . 41–43
Localization keywords . . . . . 197
Localization of orbitals
at end of calculation . . . . . 64, 226
for LMP2 . . . . . 226
with Boys method . . . . . 64
input keywords for . . . . . 172, 173, 197, 198
with Pipek-Mezey method . . . . . 42, 64
input keywords for . . . . . 172–173, 197–198
loclmp2 program . . . . . 226
locpost program . . . . . 226
log file . . . . . 7, 32, 33, 122–123
description and format . . . . . 122–123


# M


machid program . . . . . 227
MacroModel .dat files . . . . . 21
reading from Jaguar interface . . . . . 273
mass input type . . . . . 216
Masses
for frequency calculations . . . . . 53
input keyword for . . . . . 167–168
setting in atomic section . . . . . 214–216
Maximum number of iterations for geometry convergence 67, 97, 131
input keyword for . . . . . 178
Maximum number of SCF iterations . . . . . 62, 128
input keyword for . . . . . 192
Memory
input keywords for . . . . . 209–212
troubleshooting related to . . . . . 275
Memory usage keywords . . . . . 209
Memory, disk and i/o information output option
input keyword for . . . . . 201
Memory, disk, and i/o information output option . . . . . 109
Methanol . . . . . 48
Method options . . . . . 39
Methods window . . . . . 60–65
accuracy level . . . . . 63–64
analytic corrections . . . . . 64
convergence issues . . . . . 62–63
initial guess selection in . . . . . 61–62
input keywords for . . . . . 191–195
localization of orbitals . . . . . 64
output from . . . . . 108, 116

symmetry, use of in . . . . . 65
wavefunction type selection in . . . . . .60–61
Molden orbitals file (.molf file) output option . . . . . 115
input keyword for . . . . . 204
Molecular charge . . . . . 16
input keyword for . . . . . 168
Molecular properties
calculating output from . . . . . .102–106
Molecular properties, calculating . . . . . .48–52
Molecular state keywords . . . . . 167
Molecular structure—see Geometry input or Geometry optimization
Møller-Plesset second-order perturbation theory—see MP2
MOPAC, interface from Jaguar to . . . . . .133–136
Mouse functions . . . . . 20
MP2 (Møller-Plesset second-order perturbation theory) . . 41–43, 148–152, 226
input keywords for . . . . . 171
optimization output for local MP2 . . . . . .93–97
output from . . . . . 91
MQM basis set (.bas file) output option
input keyword for . . . . . 204
mulk input type . . . . . 216
Mulliken population analysis . . . . . .51–52
ch program . . . . . 226
for basis functions . . . . . 52
output from . . . . . 104
input keyword for . . . . . 187
output from . . . . . .104–106
recalculating multipole moments from . . . . . .50, 52
output from . . . . . 106
Mulliken spin populations . . . . . 51
multip input type . . . . . 216
Multiple Jaguar jobs
running from Jaguar interface . . . . . .29–31
running with jaguar batch . . . . . .230–235
Multiplicity . . . . . 16
input keyword for . . . . . 168
output from . . . . . 89
Multiplicity keywords . . . . . 167
Multipole moments . . . . . 50, 102–103
calculating . . . . . .50, 130
ch program . . . . . 226
ESP fitting
constraining to reproduce multipole moments 49, 103, 104, 131
recalculating multipole moments from . . 49, 50, 104
input keyword for . . . . . 186
lmp2dip program . . . . . 226
Mulliken population analysis, recalculating multipole moments from . . . . . 50, 52, 106
output from . . . . . .102–103
output in atomic units . . . . . 109
tensors listed in output . . . . . 102
units in output . . . . . 202
Multipole moments in atomic units also output option . 109
Murtagh-Sargent method
for Hessian refinement
input keyword for . . . . . 180
for Hessian updating
input keyword for . . . . . 180


## N


Natural Bond Orbital calculations . . . . . 52, 229–230
NBO calculation, requesting . . . . . 161
NBO calculations . . . . . 52, 229–230
output from . . . . . 106
nbo inpu file section . . . . . 229
Neighbor ranges . . . . . 110, 244, 245–246
New geometry, inputting . . . . . .8–9
Newton-Raphson step
output of maximum and RMS for geometry optimization
input keywords for . . . . . 183
Nice option for "jaguar run" command . . . . . 157
Nitrobenzene . . . . . 48
NLDA (non-local density approximation) . . . . . .39, 39
nmder . . . . . 190
Non-local density approximation (NLDA) . . . 39, 39, 153
Nuclear repulsion energy . . . . . .86, 89, 99, 100, 116
Nuclear-point charge energy from solvation calculations 99
Nuclear-solvent energy from solvation calculations . . . 100
nude program . . . . . 226
Number of iterations for geometry convergence, maximum 67, 97, 131
input keyword for . . . . . 178
Numeric updating of Hessian
input keyword for . . . . . 180
Numerical gradient of energy . . . . . 66
input keywords for . . . . . .178, 182
output from . . . . . .97–98
Numerical Hessian, printing in output . . . . . 202
Numerical methods . . . . . 64, 139–142
.cutoff file determination of . . . . . 255
input keyword for . . . . . 193
Numerical second derivative of energy . . . . . 53
input keywords for . . . . . .182, 189
nude program . . . . . 226
output from . . . . . .106–107
updating
input keyword for . . . . . 180


## O


OCBSE convergence scheme . . . . . 62
output from . . . . . 108
onee program . . . . . 225
output from . . . . . 86, 95, 97
One-electron Hamiltonian output option . . . . . .110–111
input keyword for . . . . . 202
One-electron integrals . . . . . 86
calculation with program onee . . . . . 225
energy contribution listed in output . . . . . .89, 116
for solvation . . . . . 98, 100, 116
Open shell singlet . . . . . 197
Open-shell systems . . . . . 60–61, 192

Optimization window . . . . . .66–79
output from . . . . . 93–98, 102
Optimizing
transition states . . . . . 72
Optimizing geometry . . . . . .66–79
calculating forces only . . . . . 66
der1a and der1b programs . . . . . 226
input keyword for . . . . . 177
constraining bond lengths or angles .14, 69–70, 70–71, 164
input keywords for . . . . . 179
output from . . . . . 94
constraining Cartesian coordinates . . . 10–11, 70–71
convergence criteria . . . . . 68, 96, 97
gradient-related only . . . . . 179
input keywords for . . . . . 178, 181–183
convergence criterion for SCF . . . . . 67
detailed output . . . . . 110
fixed bond lengths or angles for . . . . . .14, 164
input keywords for . . . . . 179
output from . . . . . 94
fixed Cartesian coordinates for . . . . . .10–11
frozen bond lengths or angles for . . . . 69–70, 70–71
frozen Cartesian coordinates for . . . . . .70–71
GDIIS method
input keyword for . . . . . 179
generating input with new geometry . . . . . 130
geopt program . . . . . 226
in solution . . . . . 46
output from . . . . . 102
without gas phase calculation . . . . . .48, 177
initial Hessians for . . . 15–16, 68, 77–78, 106, 134–135, 163–164
input file section . . . . . 161, 219–220
input keyword for . . . . . 179
input keywords for . . . . . .177–183
limiting step size for . . . . . .69, 185
input keywords for . . . . . .181, 182
maximum number of iterations for convergence . . 67, 97, 131
input keyword for . . . . . 178
output from . . . . . 93–98, 102
bond lengths and angles . . . . . 94, 109–110
forces table . . . . . 94
refinement of initial Hessian for 15–16, 76–77, 77–78, 163–164, 180
tips . . . . . .130–131
troubleshooting . . . . . 274
trust radius for . . . . . .69, 185
input keywords for . . . . . .181, 182
updating of Hessian during
input keyword for . . . . . 180
Orbital energy . . . . . 89
Orbital output keywords . . . . . 205
Orbital reordering . . . . . 161
Orbitals
combining using orbman section . . . . . .223–224
information in output .86–87, 88, 95, 97, 116, 118–122
GVB orbitals . . . . . .92, 118, 119, 120, 206
input keywords for printing . . . . . 205–207, 221
Orbitals output options
How: . . . . . 119–122, 221
What: . . . . . 119
When: . . . . . .118–119
plotting with plot section . . . . . .228–229
reordering using orbman section . . . . . .223–224
Orbitals output options . . . . . .118–122
How . . . . . 119
input keywords corresponding to . . . . . .205–207
printing orbitals—How: . . . . . 119–122, 221
input keywords for . . . . . .206–207
printing orbitals—What: . . . . . 119
all orbitals . . . . . 119, 120, 221
GVB orbitals . . . . . .119, 120
input keywords for . . . . . 206
occupied orbitals . . . . . 119, 120, 221
printing orbitals—When: . . . . . .118–119
after final localization . . . . . 119
after GVB initial guess is generated . . . . . 118
after HF initial guess is generated . . . . .118, 120
after SCF iterations . . . . . 119
at end of job . . . . . 119
each iteration (in AO space) . . . . . 118
each iteration (in CO space) . . . . . .118, 120
input keywords for . . . . . 206
What . . . . . 119
When . . . . . 118
orbman input file section . . . . . 223
Order of Jaguar programs run, specifying . . 161, 225–228
Organometallics
improving convergence . . . . . 61, 128–130
Output file . . . . . 7, 26, 28, 32, 85–122, 123
echoing input file in output . . . . . 225
effect of calculation options on content . . . . . .90–108
Files output settings . . . . . .114–115
general description . . . . . .85–108
Orbitals output settings . . . . . .118–122
Per Iteration output settings . . . . . .116–117
Standard output settings . . . . . .108–114
summarizing . . . . . .80–85
viewing in interface window . . . . . .32, 123
Output file changes
basis set calculations . . . . . 108
DFT calculation options . . . . . 90
frequency, IR, and thermochemistry calculations . 106
geometry and transition state optimizations . . . . . 93
geometry optimizations in solution . . . . . 102
GVB calculations . . . . . 91
GVB-RCI calculations . . . . . 93
LMP2 calculation options . . . . . 91
methods other than DIIS . . . . . 108
non-default properties selections . . . . . 102
optimizations with GVB-RCI wavefunctions . . . . . 97
solvation calculations . . . . . 98
Output options . . . . . .108–122

Files output options . . . . . 114–115
input keywords corresponding to . . . . . 203–204
Orbitals output options . . . . . 118–122
input keywords corresponding to . . . . . 205–207
Per Iteration output options . . . . . 116–117
input keywords corresponding to . . . . . 204–205
Standard output options . . . . . 108–114
bond lengths and angles . . . . . 109
connectivity table . . . . . 110
detailed timing information . . . . . 109
echo input file and parameter list . . . . . 108
Gaussian function list (basis set) . . . . . 111
Gaussian function list (derivatives) . . . . . 114
geometries in atomic units also . . . . . 109
geometry optimization details . . . . . 110
input keywords corresponding to . . . . . 201–202
memory, disk, and i/o information . . . . . 109
multipole moments in atomic units also . . . . 109
one-electron Hamiltonian . . . . . 110
overlap matrix . . . . . 110
Output, summarizing . . . . . 80–85
Output—see Output file, Output options, Standard output options, Files output options, Per Iteration output options, Orbitals output, Babel
Overlap matrix
eigenvector and eigenvalue output option
input keyword for . . . . . 202
in DIIS error vector . . . . . 89, 123
smallest eigenvalue, listed in output . . . . . 86
Overlap matrix output option . . . . . 110
input keyword for . . . . . 202


# P


Parallel
jobs that can't be run in . . . . . 286
running jobs in . . . . . 27, 285
Parallel Jaguar module
about . . . . . 277
installing . . . . . 277
supported platforms . . . . . 277
system requirements
HP . . . . . 284
IBM . . . . . 283
SGI . . . . . 278
Sun . . . . . 284
path input file section . . . . . 225
Path specifying order of programs . . . . . 161
input file section for . . . . . 225–228
pbf program . . . . . 183, 226
output from geometry optimizations . . . . . 102
Per Iteration output options . . . . . 116–117
All J and K matrices, AO space . . . . . 117
Coulomb and exchange matrices output option . . 117
density matrix . . . . . 116
density matrix output option . . . . . 116–117
energy components . . . . . 116
energy components output option . . . . . 116
Fock matrix in AO (HF) or MO (GVB) space . . . 117
Fock matrix in CO space . . . . . 117
Fock matrix output options . . . . . 117
GVB data output option . . . . . 117
GVB f, a, b, ci, and other coefficients . . . . . 117
input keywords corresponding to . . . . . 204–205
perl . . . . . 155, 158
Physical constants and conversion factors
input keyword for . . . . . 193
Pipek-Mezey localization . . . . . 42, 64
input keywords for . . . . . 172–173, 197–198
orbital printing . . . . . 119
pKa calculation
ab initio . . . . . 291
conformational flexibility in . . . . . 295
empirical corrections in . . . . . 294
equivalent sites in . . . . . 295, 297
geometry optimization in . . . . . 292
initial geometry in . . . . . 310
input files for . . . . . 308
monitoring . . . . . 310
multiple protonation sites in . . . . . 295, 297
running . . . . . 309
single point energies in . . . . . 292
solvation free energy in . . . . . 292
theory . . . . . 291
pKa prediction module
installing . . . . . 308
introduction . . . . . 289
parameterization results . . . . . 300
parameterized functional groups . . . . . 299
results . . . . . 298, 300–307
Plot data, generating . . . . . 161
plot input file section . . . . . 228
Point charges, input file section for . . . . . 161, 222
pointch input file section . . . . . 222
Poisson-Boltzmann equations . . . . . 47
Poisson-Boltzmann solver . . . . . 46–48
polar program . . . . . 226
Polarizability . . . . . 50–51, 187, 226
input keywords for . . . . . 186, 187, 188
output from . . . . . 104
post program . . . . . 227
output from . . . . . 99
for solvation . . . . . 101
Post-SCF DFT calculations . . . . . 38
Potential
plotting with plot section . . . . . 228–229
Potential energy surface scan . . . . . 78–79
Powell method
for Hessian refinement
input keyword for . . . . . 180
for Hessian updating
input keyword for . . . . . 180
pre program . . . . . 86, 225
output from . . . . . 86
for geometry optimization . . . . . 94
for solvation . . . . . 98
Pressure for thermochemical calculations . . . . . 55
input keyword for . . . . . 189

output . . . . . 107
P-RFO level shifting
input keyword for . . . . . 180
probe program . . . . . 87, 225
Probe radius of solvent . . . . . 47, 324–325
Program order, specifying . . . . . 161, 225–228
Programs in Jaguar—see relevant program name or Jaguar programs
Properties keywords . . . . . 185
Properties window . . . . . 48–52
electron density . . . . . 51
ESP charge fitting . . . . . 49–50
Mulliken population analysis . . . . . 51–52
multipole moments . . . . . 50
output from . . . . . 102–106
polarizability and hyperpolarizability . . . . . 50–51
Pseudospectral method, explanation of . . . . . 139–142
publications, citing Jaguar in . . . . . 3

# Q

QST-guided transition state searches . . . . . 72–74
additional structures for . . . . . 164, 165
input keyword for . . . . . 177
LMP2 delocalization for . . . . . 172, 214
Quadratic energy error
input keyword for output . . . . . 202
Quadratic synchronous transit—see QST-guided transition state searches
Quit option . . . . . 7, 33

# R

Radian units for geometry input . . . . . 167
RCI (restricted configuration interaction) calculations . 44–45, 146–148, 212
optimization output . . . . . 97–98
output from . . . . . 91, 93
rci program . . . . . 226
rci program . . . . . 226
Read window . . . . . 16–19
Reading input files . . . . . 16–19
file types . . . . . 17–19, 273
reading geometry input . . . . . 16–19
troubleshooting . . . . . 273
Realigning and recentering geometry . . . . . 23
Reorganization energy . . . . . 101
Reset option . . . . . 33
Resolution . . . . . 22
RESP file output option
input keyword for . . . . . 204
Restarting jobs . . 130, 131, 131–133, 185, 203, 205, 213
Restricted configuration interaction—see RCI calculations
Restricted open-shell wavefunctions . . . . . 60–61
input keyword for . . . . . 192
Results, reporting
final . . . . . 81
for each atom . . . . . 84
intermediate . . . . . 83
RFO level shifting
input keyword for . . . . . 180
RODFT—see Restricted open-shell wavefunctions
ROHF—see Restricted open-shell wavefunctions
Run window . . . . . 6–7, 26–28, 124, 125, 156
Running jobs . . . . . 6–7, 26–32
by hand (outside the interface) . . . . . 156–158
by hand (outside the interface), . . . . . 26
from the interface . . . . . 6–7, 26–28
troubleshooting . . . . . 266–272
rwr program . . . . . 87, 226
output from . . . . . 94, 95, 97

# S

Sample calculation . . . . . 5–7
Save window . . . . . 31–32, 33, 158, 160
Scaling frequencies . . . . . 54
input keywords for . . . . . 189
Scanning . . . . . 78–79, 230
SCF energy output . . . . . 88–90
SCF iteration keywords . . . . . 204
SCF iterations maximum number of
input keyword for . . . . . 192
SCF iterations, maximum number of . . . . . 62, 128
SCF level shift . . . . . 63
SCF method keywords . . . . . 191
scf program . . . . . 226
HF then GVB . . . . . 93
output from . . . . . 93
output from . . . . . 87–90, 92, 95, 97, 116–117
for solvation . . . . . 98, 99–101, 116
use in solvation calculations . . . . . 98
SCHRODINGER directory . . . . . 267, 268
SCHRODINGER environmental variable . . . . . 267, 268
SCRF method for solvation calculations . . . . . 46–47, 98
energy output . . . . . 101
output from . . . . . 98–102
Search method, transition state . . . . . 72
Searching
along paths and eigenvectors . . . . . 75
Second derivative of energy
input keyword for updating numerical calculation of . 180
input keywords for numerical calculation of . 182, 189
nude program . . . . . 226
numerical calculation of
output from . . . . . 106–107
Self-consistent reaction field method for solvation calculations . . . . . 46–47, 98
energy output . . . . . 101
output from . . . . . 98–102
Shells
for grids . . . . . 250–252
information in output . . . . . 86, 88, 92, 111–114
Simons' method for trust radius adjustment
input keyword for . . . . . 181
Singlet, open shell . . . . . 197
sole program . . . . . 227
output from . . . . . 101

Solute cavity energy . . . . . .99, 101
Solution phase energy from solvation calculations . . .100–101
solv program . . . . . 226
output from . . . . . 99
Solvation . . . . . 46–48, 98–102
choosing van der Waals radii for . . . . . 98
from .lewis file . . . . . .255–265
input file sections for . . . . . 161, 214–217
input keyword for . . . . . 184
convergence criterion for . . . . . 101
input keyword for . . . . . 185
dsolv program . . . . . 227
energy output . . . . . 98–101, 116
electron-nuclear energy . . . . . 100
first shell correction factor . . . . . 101
nuclear-solvent energy . . . . . 100
reorganization energy . . . . . 101
solute cavity energy . . . . . .99, 101
solution phase energy . . . . . .100–101
solvation energy . . . . . .101, 116
total solvent energy . . . . . 100
input keywords for . . . . . .183–185
molecular surface for . . . . . 98
output from . . . . . 98–102, 116
restarting calculation with new input file . . . . . 185
sole program . . . . . 227
solv program . . . . . 226
solvent choice for . . . . . .47–48
input keywords for . . . . . .184–185
Solvation energy . . . . . .101, 116
Solvation energy convergence criterion
input keyword for . . . . . 185
Solvation keywords . . . . . 183
Solvation window . . . . . .46–48
output from . . . . . .98–102
Solvent choice for solvation calculations . . . . . .47–48
input keywords for . . . . . .184–185
Solvent parameters . . . . . 47
SPARTAN
archive files, generating with Jaguar . . . . . 115
input keyword for . . . . . 204
Spectral information . . . . . .54–55
input keyword for . . . . . 189
output . . . . . 107
Spin multiplicity . . . . . 16
input keyword for . . . . . 168
output from . . . . . 89
Spin populations, Mulliken . . . . . 51
SQM frequency scaling method . . . . . 54
Stage . . . . . 38
Standard output keywords . . . . . 201
Standard output options . . . . . 94, 108–114
bond lengths and angles output option . . . . .109–110
connectivity table output option . . . . . 110
detailed timing information output option . . . . . 109
echo input file and parameter list . . . . . .108–109
Gaussian function list in output
for basis set . . . . . .111–114
for derivatives of basis functions . . . . . 114
geometries in atomic units also output option . . . . 109
geometry optimization details output option . . . . . 110
input keywords corresponding to . . . . . .201–202
memory, disk, and i/o information output option . 109
multipole moments in atomic units also output option 109
one-electron Hamiltonian output option . . . .110–111
overlap matrix output option . . . . . 110
State window . . . . . 16
Structure, input—see Geometry input
Structure, optimizing—see Geometry optimization
Submitting jobs—see Running jobs
Summarizing results . . . . . .80–85
Superblocks . . . . . 195
Symmetrizing geometry input . . . . . 25–26, 160, 273
finding point group . . . . . 25
tolerance . . . . . 25
Symmetry point group, reported in output . . . . . 86
Symmetry, use of in calculation 24–26, 65, 86, 225, 273
input keywords for . . . . . .168–169
specifying for GAUSSIAN 92 input . . . . . 137
Symmetry-related keywords . . . . . 168
Synchronous transit quasi-Newton methods—see QST-guided transition state searches
System properties, calculating . . . . . .48–52

# T

Technical support . . . . . 3
Temp directory . . . . . 27, 28, 86, 158, 159, 272
errors related to . . . . . .270–271
jaguar.hosts specification of . . . . . 124, 125, 126
Temperatures for thermochemical calculations . . . . .55–56
input keywords for . . . . . .189–190
output . . . . . 107
Temporary directory
saving at end of job . . . . . 157
Temporary directory—see Temp directory
Temporary files . . . . . 28
saving at end of job . . . . . 157
Thermochemical properties . . . . . .55–56
input keywords for . . . . . .189–190
output . . . . . 107
Time stamps to log file option for "jaguar run" command . 157
timex program . . . . . 227
Timing . . . . . 109
Total solvent energy from solvation calculations . . . . . 100
Transition metals . . . . . 63
improving convergence . . . . . 61, 128–130
initial guess for . . . . . 61
Transition state keywords . . . . . 177
Transition state optimization . . . . . .66–79
constraining bond lengths or angles . . . . . .69–70
input keywords for . . . . . 179

convergence criteria . . . . . 68
gradient-related only . . . . . 179
input keywords for . . . . . 178, 181–183
convergence criterion for SCF . . . . . 67
eigenvector following in . . . . . 76
input keywords for . . . . . 181
fixed bond lengths or angles for
input keywords for . . . . . 179
frozen bond lengths or angles for . . . . . .69–70
in solution . . . . . 46
without gas phase calculation . . . . . .48, 177
initial Hessians for . . . 15–16, 68, 77–78, 134–135, 163–164
input keyword for . . . . . 179
input keywords for . . . . . .177–183
level shifting of Hessian
input keyword for . . . . . 180
limiting step size for . . . . . 69
input keywords for . . . . . .181, 182
maximum number of iterations for convergence . . . 67
input keyword for . . . . . 178
output from . . . . . .93–98
refinement of initial Hessian for 15–16, 76–77, 77–78, 163–164, 180
trust radius for . . . . . 69
input keywords for . . . . . .181, 182
updating of Hessian during
input keyword for . . . . . 180
Transition state search method . . . . . 72
Trial wavefunction—see Initial guess
Troubleshooting . . . . . .266–275
Trust radius for optimizations . . . . . .69, 185
adjustment during optimization . . . . . 69
input keywords for . . . . . .181, 182
application of during optimization
input keyword for . . . . . 181
input keywords for . . . . . .181, 182
twm window manager . . . . . 127
Two-electron integrals . . . . . 64
energy contribution listed in output . . . . 89, 108, 116
breakdown into J and K parts . . . . . .92, 116


## U


UDFT—see Unrestricted wavefunctions
UHF—see Unrestricted wavefunctions
Uncontracted basis functions . . . . . .111, 112
Unrestricted wavefunctions . . . . . .60–61
input keyword for . . . . . 192


## V


van der Waals radii . . . . . 50
for solvation calculations . . . . . 98
input file sections for . . . . . 161, 214–217
input keyword for . . . . . 184
listed in output . . . . . 98
setting from Lewis file data . . . . . 260
van der Waals surface . . . . . 50
Variables in geometry input 10, 13, 14, 70, 71–72, 164–165
vdw input type . . . . . 216
vdw2 input type . . . . . 216
Vibrational frequencies . . . . . 53
freq program . . . . . 226
input keywords for . . . . . .182, 189
output . . . . . .106–107
scaling . . . . . 54
input keywords for . . . . . 189
View File button . . . . . 32, 33, 123
virial ratio . . . . . 201
Virtual orbitals
number in output . . . . . 205
output . . . . . .89, 119


## W


Water . . . . . 48
Wavefunction type . . . . . 60
What: Orbitals output options . . . . . 119
all orbitals . . . . . 119, 120, 221
GVB orbitals . . . . . .119, 120
input keywords for . . . . . 206
occupied orbitals . . . . . 119, 120, 221
When: Orbitals output options . . . . . .118–119
after Boys localization
input keyword for . . . . . 206
after final localization . . . . . 119
after GVB initial guess is generated . . . . . 118
input keyword for . . . . . 206
after HF initial guess is generated . . . . . .118, 120
input keyword for . . . . . 206
after SCF iterations . . . . . 119
input keyword for . . . . . 206
at end of job . . . . . 119
each iteration (in AO space) . . . . . 118
input keyword for . . . . . 206
each iteration (in CO space) . . . . . .118, 120
input keyword for . . . . . 206
input keywords for . . . . . 206


## X


XYZ file (.xyz file) output option . . . . . 115–??


## Z


Zero point energies—see Thermochemical properties
zmat input file section . . . . . 162
zmat2 input file section . . . . . 162
zmat3 input file section . . . . . 162
Z-matrix format . . . . . 8, 11–14, 160, 163–165
dummy atoms in . . . . . 14
variables in . . . . . 13, 14, 164–165
zvar input file section . . . . . 164
zvar2 input file section . . . . . 164
zvar3 input file section . . . . . 164

# Keywords


## B

babel . . . . . 199–201
babelg . . . . . 199–201
basis. . . . . 190–191, 221

## C

cfiterr. . . . . 187–188
covfac . . . . . 110, 167

## D

dcoarse . . . . . 209
dconv. . . . . 183, 187, 191
denspc . . . . . 187, 188
dfine . . . . . 209
dgrad . . . . . 209
dmedium . . . . . 209
dufine . . . . . 209

## E

econv. . . . . 183, 187, 191
efield . . . . . 187, 188
epsin . . . . . 184
epsout . . . . . 184
esolv0 . . . . . 185

## F

freqfrag . . . . . 219

## G

gcharge . . . . . 188, 209, 254
gcoarse . . . . . 208, 254
gconv1-gconv7 . . . . . 182–183
gcphf . . . . . 208
gdftfine . . . . . 207
gdftgrad. . . . . 207, 254
gdftmed. . . . . 207, 254
geldens . . . . . 187, 188, 208, 254
gfine . . . . . 208, 254
ggrad . . . . . 208, 254
gmedium . . . . . 208, 254
gmp2 . . . . . 208
gufine . . . . . 208, 254

## I

iacc . . . . . 193
iaccg . . . . . 178, 181
iacscf . . . . . 193
icavity . . . . . 183
icfit . . . . . 185
ichange . . . . . 195
iconv . . . . . 192
icpfrag . . . . . 219
idelfrag . . . . . 219
idelocv. . . . . 172
idenavg . . . . . 176, 194
idft. . . . . 173–176
idoabe . . . . . 169
ifollow. . . . . 181
ifreq. . . . . 189
igeopt. . . . . 177
igonly . . . . . 136, 196
igrado . . . . . 179
iguess. . . . . 196–197, 220
igvball . . . . . 169, 170
igvbsel. . . . . 169, 170, 171
ihamtyp . . . . . 197
iheter . . . . . 171
ihfgvb . . . . . 169, 196–197
ihuptyp . . . . . 180
ilagr. . . . . 179
incdip. . . . . 185, 187
inhess. . . . . 179, 180, 219
intopt. . . . . 165, 178
iordboy . . . . . 198
ip1, ip3-ip8, ip11-ip13, ip18-ip19, ip24-ip26, ip170, ip173, ip192-ip193. . . . . 201–202
ip100 . . . . . 206
ip100-101, ip103-107 . . . . . 206
ip105 . . . . . 206, 207
ip107 . . . . . 199
ip15, ip17, ip110, ip121-ip123, ip149, ip188, ip201 . . . 205
ip151 . . . . . 133, 203
ip152 . . . . . 133, 205
ip160 . . . . . 137, 138, 203–204
ip165 . . . . . 203
ip170 . . . . . 91, 202
ip28 . . . . . 190
ip90, ip160-ip161, ip163-ip165, ip168, ip172, ip175. . . 204
ipolar. . . . . 186, 187, 188
ipopsym. . . . . 169
ipvirt . . . . . 205
iqcoarse. . . . . 211–212
iqfine. . . . . 211–212
iqgrad . . . . . 211–212
iqmedium . . . . . 211–212
iqst. . . . . 172, 177

iqufine . . . . . 211–212
irder . . . . . 189
irefhup . . . . . 180
ireson . . . . . 172
irfo . . . . . 180
isolv . . . . . 183
isolvg . . . . . 184
isqm . . . . . 189
istavg . . . . . 194
isurf . . . . . 184, 265
isymm . . . . . 168, 187
iteravg . . . . . 194
itradj . . . . . 181, 182
itrcut . . . . . 181, 182
itrvec . . . . . 181
itwice . . . . . 195
iuhf . . . . . 192
iunit . . . . . 162, 163, 167, 222
ivanset . . . . . 184, 265
ixtrboy . . . . . 198, 198–199

## J

jdft . . . . . 176
jksep . . . . . 193

## K

kesep . . . . . 184, 201

## L

lastwv . . . . . 194
lcoarse . . . . . 208
ldens . . . . . 187, 188
ldips . . . . . 186
lewdot . . . . . 169, 170, 172
lewstr . . . . . 169, 170, 184, 265
lfine . . . . . 208
lgrad . . . . . 208
lmedium . . . . . 208
loclmp2c . . . . . 172
loclmp2v . . . . . 173
locpostc . . . . . 197
locpostv . . . . . 198
lufine . . . . . 208

## M

massav . . . . . 168
maxdiis . . . . . 191
maxit . . . . . 192
maxitg . . . . . 178
molchg . . . . . 168, 222
mp2 . . . . . 171
mulken . . . . . 187
multip . . . . . 168
mxpage . . . . . 210
mxpr . . . . . 210
mxrwr . . . . . 211
mxstrip . . . . . 210

## N

nbcmax . . . . . 210
nbuck . . . . . 210
ndisk . . . . . 210
newcon . . . . . 193
nhesref . . . . . 180
nmder . . . . . 53, 178
noatcor . . . . . 193
noauto . . . . . 194
nogas . . . . . 177
nogdiis . . . . . 179
noopta . . . . . 179
nooptr . . . . . 179
nooptt . . . . . 179
nops . . . . . 194, 255
nosuper . . . . . 195
noupdat . . . . . 194
ntemp . . . . . 190
numd . . . . . 57, 191

## P

pertnd . . . . . 178, 182
press . . . . . 189

## R

radprb . . . . . 185

## S

scalfr . . . . . 189
sconv . . . . . 185
stdiis . . . . . 191

## T

tmpini . . . . . 189
tmpstp . . . . . 190
tradmn . . . . . 182
tradmx . . . . . 182
tremx . . . . . 182
treok . . . . . 182
trescal . . . . . 182
trgmx . . . . . 182
trust . . . . . 182, 185

## V

vshift . . . . . 176, 192
vshift, . . . . . 193

## W

wispc . . . . . 187, 188

## X

xcorl1 . . . . . 175–176
xcorl2 . . . . . 175–176
xcorl3 . . . . . 175–176
xcorl4 . . . . . 175–176

xcornl1 . . . . . 175–176
xcornl2 . . . . . 175–176
xcornl4 . . . . . 175–176
xexl1 . . . . . 175–176
xexl9 . . . . . 175–176
xexnl1 . . . . . 175–176
xexnl4 . . . . . 175–176
xhf . . . . . 175–176

# Y

ycorl1 . . . . . 176
ycorl2 . . . . . 176
ycorl3 . . . . . 176
ycorl4 . . . . . 176
ycornl1 . . . . . 176
ycornl2 . . . . . 176
ycornl4 . . . . . 176
yexl1 . . . . . 176
yexl9 . . . . . 176
yexnl1 . . . . . 176
yexnl4 . . . . . 176
yhf . . . . . 176

# Z

zmpmem . . . . . 211